51CTO.com
技术成就梦想
我 们 只 谈 开 发

开发月刊

Development Monthly

2013年04月号
副刊

敏捷

专访程显峰：

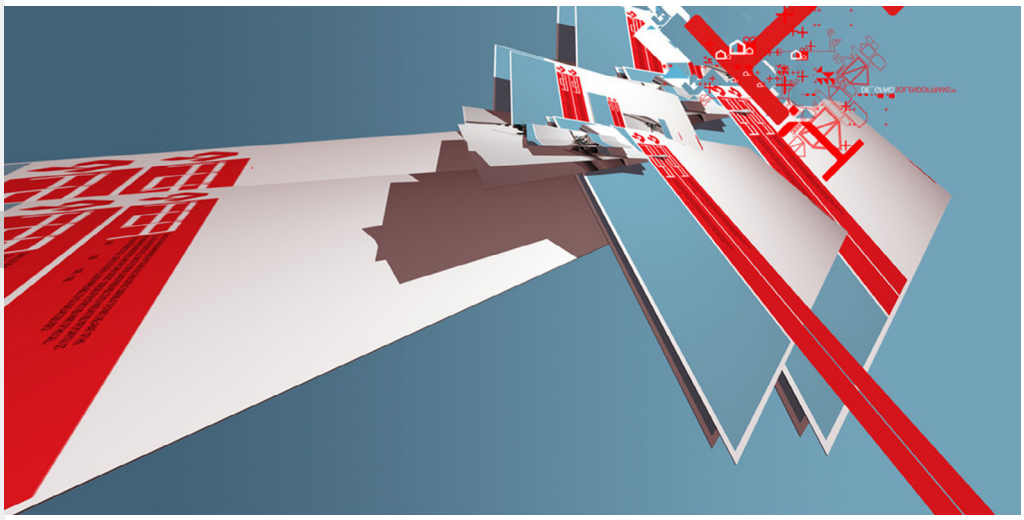
敏捷在团队中的实践与发展现状

专访陈勇：深入解读敏捷在团队中的实践

51CTO记者采访了具有17年软件研发、管理咨询经验，擅长在实际环境中应用敏捷开发实践的专家陈勇老师，为网友们解读敏捷在开发实践及管理中的一些问题。

51CTO.com
技术成就梦想

开发
月刊



主编的话

开发月刊推出已经有2年了，相信很多51CTO的网友都阅读过。

开发月刊给大家带来了一些热点技术，观点，还有一些业界新闻的线下阅读平台。发布至今，也收到了很多网友们的好评，当然也有批评。

如今，我们收集了一些网友们的建议，把开发月刊进行改版，让网友们阅读的过程得到更好的体验，当然最重要的是每隔段时间会给网友们分享最新推出的副刊。

也许有网友会问什么是副刊？副刊其实就是开发月刊主打原创内容，主要包括：专家采访，观察原创，外电译文等。我们会关注近期的热点，给网友们推出更好，更有价值的内容，同时也希望得到网友的支持！

祝好！

51CTO开发频道寄语

出版方：
北京无忧创想信息技术有限公司
责任编辑：
林师授
页面设计：
林师授
联系方法：
邮箱：linss@51cto.com
电话：010-68476606-8123
出版日期：2013年04月18日
欢迎来稿，请发送邮件至
linss@51cto.com

原创专访

外电译文

专访陈勇：深入解读敏捷在团队中的实践

■ 51CTO记者采访了具有17年软件研发、管理咨询经验，擅长在实际环境中应用敏捷开发实践的专家陈勇老师，为网友们解读敏捷在开发实践及管理中的一些问题。

有人说，实践敏捷的根本不在于敏捷本身，而在于理解敏捷背后拥抱变化的基因。确实，使用敏捷，那么你就应该知道为何敏捷。如今，国内很多敏捷团队没有真正的了解敏捷的精髓，也说明了为什么在使用敏捷的过程中并没有真正的“敏捷”。所以，不但不能真正的解决传统开发的一些问题，反而新增加了更多的问题。因此，51CTO记者采访了具有17年软件研发、管理咨询经验，擅长在实际环境中应用敏捷开发实践的专家陈勇老师，为网友们解读敏捷在开发实践及管理中的一些问题。

陈勇，17年软件研发、管理及咨询经验，擅长在实际环境中应用敏捷开发实践。具有丰富的工程技术与项目管理实践经验，从其程序员、项目经理、CMMI/FPA功能点估算/敏捷咨询师、事业部总监、副总经理等各种技术与管理岗位获得的一手经验，令其可以站在企业管理者的角度，以更广的视角来理解敏捷开发，并能配合和推动非研发部门协作推广敏捷。

工作经历：

曾以技术骨干和项目经理等身份，组织和承担开发了国庆50周年直升机编队指挥系统、空军一基地GPS数据源系统、清华同方CCTV数字电视条件接收系统、航空材料研究院无损检测系统等项目，并在其中某些项目中实践敏捷。

曾在清华同方、普天集团、亚信科技等企业担任EPG骨干、组长；曾在斯福泰克、DNV ITGS等机构担任CMMI/敏捷咨询师。

曾在中国系统与软件改进年会 、中国软件技术大会、敏捷中国大会、MPD等国际国内会议从事敏捷演讲、翻译或主持工作。

在任泰克赛尔软件公司中国部门的咨询总监、ALM事业部总监、副总经理期间，主管敏捷研发管理工具的市场、销售、支持与咨询活动，在盛大、金山、腾讯、汉王科技等知名企业深入推动其工具应用与实施活动。

当前正在作为产品经理、架构师带领一个小型团队，从事“火星人敏捷开发在线服务”的研发工作。很多课程与咨询中的最佳实践，均来自于其之前及当前参与的实际项目的一线实践。

以下是采访内容：

记者： 在使用敏捷当中，很多人都是看中了它的成本控制，能够降低在项目开发中的浪费，但是往往适得其反，不但没有降低，反而增加了开发成本，那么在成本控制这块您怎么去理解？

陈勇：看看清楚敏捷开发对成本的控制，先要看之前的方法有何问题，比如瀑布模型。

对瀑布模型而言，有两个地方控制不好成本：

一，瀑布模型中，如果项目签的需求太多而价格太低，往往在中后期会通过工时统计等发现入不敷出，但由于普通模型这时候正处在编码末期或测试前期，因此很难就此停下，或删除某些次要功能继续开发（因为在晚期删除功能往往成本很高）。

对这个问题，敏捷开发解决得比较不错，也很容易实施。渐进式交付使得“随时”停止开发并进行发布变得可能。

但要注意如果优先级排序不当，也可能会面临“重要的功能还没有开发，次要功能却开发了不少”的尴尬境地。

另一个问题是提前交付可以换回一些款项，但并不能彻底解决成本超支的问题。这时候应该配合类似功能点分析这类的方法来解决。也就是早期用功能点报价；中间如果通过计数发现由于需求蔓延、变更等原因导致需求的总量上升了，那么乙方有权要求追加款项，或删除某些次要功能。这种听起来不可思议的想法，在某些国家如芬兰、澳大利亚都是很普遍的。可能我们会抱怨说甲方不赞同或不理解这种过于“偏技术”方法，但如果连乙方都不赞同不理解，就不能单方面责怪甲方了。2012年底中国刚刚推出了基于功能点的软件成本估算相关的国标，试点领域是政府软件开发。虽然现在还没有对阶段性付款提出方案，可以已经可以看出未来的大趋势一定如此。

二，瀑布模型中，如果需求变更比较频繁，初期付出众多工作量的需求、设计常常被不断返工，进而又引发编码返工，造成成本上升。

敏捷开发解决这个问题的方法很简单：减少早期对需求、设计的投入，尽量早让产品“可运行”以便让客户提出反馈意见。

不过这个方法极其容易被“用过头”，比如有些团队可能放弃了必要的需求和设计，导致由于蛮力编码造成编码混乱而不断返工，整个产品则像一个没有骨架的沙雕一样难以做大。

正确的方法是：在渐进式开发、渐进式交付的同时，做渐进式需求和设计。我们要避免的是需求和设计返工，而不是需求和设计活动本身。因此对大中型产品而言，应该保证每个阶段的编码之前都有简单设计用以保证编码的正确性；而被客户确认后的功能至少要“后补”需求和设计，以备未来回溯。这样既避免蛮力编码返工，又能避免推翻设计的返工。

记者：做到敏捷开发，每个团队都要经历一个转型期，那么在转型期还需要每个团队根据自身的不同，找出合理有效的解决方法。一般如何去处理这个问题？

陈勇：这是培训和咨询过程中最常见的一个问题，也是最难回答的。有几个步骤可以帮助找出合理有效并适合团队的方法。

一、对有效性进行定义和评价

完全量化管理可能做不到，但“半年后，我们希望能每月发一个可交付版本”或“把总体测试时间降低一半”这类明确个别定义还是可以找到的。当然不同团队首批目标不同，首批要实践的内容也不同。但有了目标，就可以防止团队借口“项目特色”，言敏捷之名行混沌之事。

二、阶段性检查和推进目标

前几个迭代往往离目标很远，很容易忙了半天才发现走偏了。这时候可以设置某些临时目标来作为“路标”。比如，假设某团队经常处于“很多活都在干，但没有一个干完了，因此也不能交付”的状态，而几个月后希望做到“当月计划功能的交付比例达到80%”，那么可能需要这样几个路标（实际工作不只如此）：

第一个月：实现任务的看板管理

第二个月：实现基于故事的看板管理

第三个月：当月计划功能交付比例达到50%

第四个月：控制在制品数量在2N−1以下（N是团队人数，此目标即“同时开工的工作量不超过2N−1个”）

……

第N个月：当月计划功能交付比例达到80%

在这个例子当中可以看到，如果一上来就直接设定“当月计划功能交付比例达到50%”之类的度量数据，可能不但很难达到，甚至可能由于没有看板管理而不能度量。

三、同时选择超过一个团队进行试点

敏捷开发的试点过程千变万化，如果只找一个团队试点会有问题。如果失败了，大家会以为敏捷不行；如果成功了，大家会照搬经验。

所以最好的方法是让若干个团队各自基于自己的目标进行试点，如果其中有多个取得成功，大家会意识到原来成功的方法有很多种，就能灵活应用在自己团队中进行实验和推广了。

四、需要对“商业目标”与“一线实践”有很深入的理解

做到这一点非常困难，然而偏偏这一点又是成败的关键。

经常见到很多团队或成员一拥而上开始试点某个局部实践，有时候是Scrum中的扑克牌估算或每日立会，有时候是XP中的自动化测试或结对编程。但做了一段时间，由于没有目标，尤其是没有那些让团队听了之后就不会放弃的目标，很快就坚持不下去了。

要培养商业目标意识很难，甚至很多从业之前没有做过高级管理职位的敏捷培训师或咨询师都不具备，更不用说普通一线工作人员了。一种做法是让高层比如部门经理来宏观协调敏捷的实施，而不要完全自下而上。高层经理的职业特性保证了他们会潜移默化地关心最终实施的效果胜过实践本身。说起来，“效果胜过实践”本身就是敏捷宣言中“可工作软件胜过繁杂文档”的现实体现。

五、要注意敏捷开发的生态系统问题

刚才提到尝试个别实践经常失败，一个原因是因为缺少目标只关注实践，另外一个则是忽略了与此实践相关的生态系统。

比如每日立会，看起来非常简单，但实践起来困难重重。比如为什么我要告诉别人我的进度？为什么我要告诉别人我遇到的困难？谁因为什么原因会提供帮助？……这些潜在的问题，都需要相应的团队模型来解决。敏捷开发生态系统就是用来描述各种活动之间的关系的，这个在我的“敏捷开发生态系统”系列博客中有较为详细的描述。

记者：敏捷过程中有一个我认为最重要的部分是需求拆分，您能谈谈如何进行合理的拆分吗？

陈勇：需求拆分是个业界难题，在敏捷开发领域是如此。尽管每个敏捷流派对这个问题说起来都头头是道，但是要说谁有一个过目不忘的好方法，还真没有。

真正解决需求拆分问题的关键有两个：拆分后的颗粒度如何控制，拆分后的需求结构如何表达。前者可能问得比较多，因为多数程序员都要关心；后者只有产品经理会关心，所以显得不太突出，但对大产品和持续开发的产品而言则是个关键问题。

一、颗粒度问题

这个问题在敏捷框架下基本无解。

但在功能点分析（Function Point Analysis）中是一个基本概念。FPA通过对软件中的“业务数

据”以及对其进行的“业务操作”为基本单元建立了颗粒度的概念，进而建立了规模的概念。这些概念的应用成熟度、推广程度、数据积累量（公开的在10000个左右，在咨询公司手中但非公开的在50000个左右，实际使用量不可估量）都远远超过敏捷开发或其他体系提出的方法，包括“故事点”。而且“业务操作”的概念非常接近用户故事，也包括如减少依赖、完整交付客户价值等描述。

下面举一个例子来看功能点到底是什么。这里将略去功能点中对数据、操作的复杂细节分析过程，而只说大概。

假设在一个OA系统中需要管理“用户”“角色”“权限”这三种业务数据，那么他们就是业务数据；而对他们分别进行的“增删改查”就是业务操作。比如“新建用户”“列出所有用户”都是业务操作。“业务”这个词在这里是相对于“技术”而言的，比如“新建用户”对客户而言是一个容易理解、认可的操作，而且客户管理员会告诉我们：

“是的，我现在每天都在新建用户，因为最近正在招人”而不会说“是的，我最近每天都在数据库表中新建记录，还会同步更新联系人数据表，因为最近正在招人”，虽然后者也的确发生了，但却不是客户日常工作的内容。

更神奇的是，“新建用户”这个操作，在无法获得更多细节的情况下，可以假设工作量为4人天（在OA系统中，其他系统有调整因子）；而统计还表明，“用户”及所有与之相关的操作如刚才说的“新建用户”以及其他“删除用户”“列出所有用户”“编辑用户”……的总工作量（从需求到部署）大约是40人天，也就是2个人月，即使暂时不能确定到底有多少个操作。而“用户”“角色”“权限”及其所有操作完成的工作量自然就是大约6人月。尽管每个数据或操作的时间可能会有出入，但总体规模则误差不大。

这得益于大数定理及众多统计数据的分析结果。本段内容的详情可在Google查询“nesma”（一个总部在荷兰的世界第二大功能点组织），这个简化体系称为“Indicative Function Point”（指示性的功能点），在其网站上有荷兰语和英语介绍。

如果对用户故事的定义（而非“业务操作”的定义，因为后者更完善、严格）略作约束，则可以迅速与功能点中的“业务操作”兼容。兼容的结果有两个，首先是很容易把握颗粒度，其次是功能点与工作量的比例关系极佳，可以迅速推知工作量。

二、需求结构问题

在敏捷开发中有一个不完善的答案：Product Backlog 和 Sprint Backlog。前者是整个产品的待开发项，后者是当前迭代的待开发项，两者内部都按优先级进行排序。

这个答案对项目经理（可以模糊理解为Scrum Master）和开发人员足够用了，因为他们主要关心时间上的开发问题，也就是现在让我做什么，什么时候要的问题。但对产品经理（可以模糊理解为Product Owner）而言，时间是一个维度，空间则是另外一个维度。

所谓空间维度，就是系统-子系统-模块-大需求-小需求……之间的关系问题，这完全不是一个一维表格能解决的。由于敏捷开发主要是一线开发人员和项目经理发明并推动的，所以空间维度一直被忽略。但这是需求分解过程中一个绕不过去的问题，因为人们不可能面对一个一个一维表格回答这

个问题：“我们所有的功能都包括在这里呢吗？哪部分还略显粗糙希望继续分解？这些大需求包含哪些小需求？”

在UML中的答案比较完善，“用户 – Use Case”（人–功能）很好地把一批相关的功能联系起来。虽然只是非常简单的表达，但比把几个毫无关联的故事按优先级排在一起，或把几个密切相关的故事相隔十万八千里放置还是进步不少。UML中的模块则是一个更宏观的需求“目录”。

FPA虽然没有需求关系表达，但如果把“业务数据 – 业务操作”作为一个树形结构，很类似把UML中的“实体 Use Case”作为一个树，形成“数据–功能”的关系。不过，这个体系没有提供更宏观尺度上的需求结构问题。

由于功能点、UML中的模块用例等概念都是有严格、确定化的定义的，所以分解的过程井然有序，很少出现“这个还要继续分吗？”“这样分是不是颗粒度太粗糙”之类的问题。我们自己正在做的项目需求就是借用FPA的业务数据–业务操作作为主要分解关系，上面再加上我们自己定义的子系统–模块两个级别形成了一个“故事树”。这个故事树比我之前所做过及所见过的任何其他方法表达的需求结构更为清晰。因此敏捷开发应该吸取或者配合这些方法，至少是其中的一部分思想来完成需求分解。

很多时候我们会觉得引入较为严格的定义会“有限制太难做”，但其实更难做的是“无规则无限制”下自由发挥。

记者：敏捷开发中需求拆分之后，任务应该怎样分工？

陈勇：即使都是使用敏捷开发，任务拆分和分工的方法也会因行业、产品的特点而有所不同。

在纯软件或技术相对单一的产品中，需求拆解到4天左右（也就是前面提到的“一个业务操作”的规模）就可以分下去或领取了，因为人们比较容易估算4天左右的工作量，也容易跟踪。在这种情况下，如果一个人同时负责几个“业务操作”，应该分解为多个任务来完成，而不是一个。因为这些业务操作可以独立完成并交付，合在一起不但不能持续交付，也会使得进展不透明。但再进一步划分没有必要，因为再划分就不能独立交付了。“独立性”是FPA中对业务操作的一个约束条件，如果按照FPA的概念来划分任务，会自动满足独立性。

在复杂环境比如嵌入式研发（涉及软件、硬件、工业设计团队）或游戏（涉及软件、美术、脚本、策划团队），需求不能直接当作任务分下去，需要做进一步的分解。这里的分解目的是把不同工种的活分配下去，对其跟踪的过程则可增进各部门配合时所需的透明度。

也有一些团队喜欢把任务分解到短至1～2小时的时间。本人没有与这样的团队进行深入探讨以了解其这样做的动机及收益是什么，但要注意“更小的任务”不等于更敏捷。我们经常说敏捷开发有利于项目透明，但要知道透明的主体是“项目外部的人”，比如产品经理、高层领导；而开发人员本人，或者小团队的若干个人之间，任务和进度本来就是透明或半透明的。因此把详细技术细节翻个底朝天并贴在白板上很可能不但于事无补反而添乱，不如把有限的管理时间用在“对外表达需求的完成情况”或“各部门的进展及需要什么相互配合”上，也就是我们之前说的两种情况。

总之，对任务的分解应该本着“需要”与“必要”的原则。不能因为想“更敏捷”而采用“更小的任务”。

记者：Sprint会议是实践scrum中最重要的事件。您认为Scrum会议在团队中应如何进行的?

陈勇：要做好计划会，应该先建好团队模型。

计划会是少有的团队齐集一堂进行的活动，无论团队模型是怎样的，都应该有助于大家以集体的力量完成计划，并完成计划的任务。

因此，一些看上去很敏捷的团队模型或做事方法，很可能是很难鼓励大家做到这一点的。比如：所有队员都是平等的；大家在会上投票取平均值作为估算结果（还有一种做法：会上不需要估算，会后领取的人自己说了算）；会后大家主动领取任务；领取的任务大家互相不干涉，自己愿意怎么做就怎么做……这里边存在什么问题呢?

第一，“所有队员平等”的团队互助的动力从哪里来？人们一定会说：都是在一起工作的同事，怎么会没有互助动力？我们曾经在一个团队实施代码审查，而且是我之前多次提到的那个最好的139团队，很快发生一次事故：一大段4000多行但包含4000多个常数的烂代码逃过了检查者的眼睛进入了代码库，并最后引发客户现场的Bug。当询问代码审查者为何让明显存在问题的代码通过检查？代码审查者很无奈地说：“我刚来不久，虽然我觉得这些代码有问题，但我感觉大家都说他（编写者）是个高手，我也就不想说什么了。其实我本人也觉得这些代码有问题”。这件事情促使我们后来形成了139团队，就是形成层次团队，由师傅来审核徒弟的代码，出问题师傅负责。这不是说所有团队一定要如此，而是说不能简单地把互助建立在“雷锋精神”上面。

第二，集体估算的目的何在？很多人认为是领导放权发扬民主，所以投票+平均值就体现了这一点。但如果一个人估10人天，另一个人估计1人天，结果到底是多少呢？总不能真的是5.5人天吧？所以估算的目的其实是想知道有没有更快的方法，有没有人在走弯路……如果为期一天的计划会耗费整个团队10多个人天，但却避免了上百人天的弯路损失，就算真正物有所值。我遇到的单个任务的最大差异是20人天比0.5人天，很可惜他们不做敏捷开发也不做估算，是任务接近尾声的时候被偶然发现的，只用了0.5人天就重写了。在一个139团队中，一般师傅要和徒弟用敏捷估算扑克估计每个任务的工作量；如果刚才这个任务出现在139团队中，师傅不会取值（20+0.5）/2，而是通过与徒弟讨论，让徒弟学会如何用0.5人天来完成这个任务，因为徒弟浪费的时间最后要算在整个师徒小组身上。如果不是139团队，也要找到类似的人们参与争论和坚持自己正确见解的动力。

第三，剩下的“主动领取任务”、“互不干涉”等，也要进行相似的分析，不能因为“看上去很自由很敏捷”就采纳。提出这些实践的团队可能与我们所处的情况差别很大，未必能拿来就用。我曾经用过“自由领取任务”，那是在10年前的一个假期里边，我邀请自己关系很近的两个老部下来帮忙做一个软件；一些零散任务就被单独拿出来，谁先做完手头工作就自由领取，其中一个人领取了8个中的5个之多。但在之后的每一个项目中，我都会先想一想，团队是否允许我再次采用这个实践。

团队模型理顺了，计划会所需的共同计划、集思广益、团队协作才能做好。

记者：在敏捷开发方法中，Scrum和XP我个人认为比较常见，您能介绍下Scrum与XP的最大区别在哪里？在实践中Scrum与XP只取其一，还是两者都取，或者全都抛弃?

陈勇：Scrum主要是关于需求和项目管理的，

而XP则主要是技术管理。

Scrum中关于Product Backlog的全部及Sprint Backlog的形成过程、评审会大部分，可以理解为需求管理内容，包括需求开发，分解，描述，排序以及进展跟踪的过程。而估算、每日立会、燃尽图等内容则属于项目管理。

XP则主要是技术管理问题，不过其结构不明确，不好分类。

这个差别也可以解释Scrum为何易于推广：因为其整个体系很清晰。比如在我的培训课程中有一个贯穿始终的练习，用三个需求的描述、分解和估算过程来演练Scrum。这个练习能从第一天上午10：00左右开始，断断续续穿插于培训中，一直持续到第二天中午；要用一个练习把所有或者只是大部分的XP实践串联起来则几乎是不可能的。另外Scrum宣贯起来也不需要大的技术革命，主要是人员职责的重新描述，部分管理活动如计划、跟踪的引进，因此门槛很低。不过反过来，由于缺少实际的技术改进，若只得其皮毛而没有利用团队、过程的变化来真正促进开发，那么Scrum失败的概率也很高。

Scrum在商业运作上也更成功。这应该归功于Scrum的发明者是项目经理或更高级别的人员，而XP则主要是一线员工，甚至说是Geek们不为过。Scrum推出了认证的Scrum Master和Product Owner两种认证，由于体系相对完整，认证课程目标听众的收入也相对较高，所以推广起来比较容易；多数咨询师更容易理解Scrum的理念而非XP的，也使之得以快速推广。

不过长远看来，团队，过程，技术三者密不可分。Scrum覆盖了前两者中的一部分，但如果没有持续集成、自动化测试等方法的支撑，迭代交付将很难完成。所以，未来应该会有并存的需要，即使不再以当前的两个名字存在。

记者：拙劣的度量指标会产生严重的后果，敏捷度量指标也是一个很有争议的问题，以您的经验如何去解决这个问题？

陈勇：有多个层次的指标可供衡量，下面从基层到高层一层层剖析一下。

一、基层度量

对最基层活动而言，一般所有度量均不用于个体考核，而只是用于改进。比如个体的缺陷率、工作完成量、按时完成率等。

之所以这样做，原因是不希望面向个体的考核破坏了团队中原本用来提高生产率的团队协作。团队协作可以消除个体错误，通过共同估算让高手的技能流向新手，等等。如果个体考核让人心有挂念，会产生问题。

不过一个不可回避的问题是：不管你们敏捷开发怎么说，但财务终究要把绩效奖金（假如有）发放到个体而非团队的账户上，怎么知道谁应该发多少呢？在一个10个队员且关系互相平行的团队里边，的确没有办法知道，因此也不得不面向个体考核，并最终不可避免地毁掉团队协作。

但在一个139团队里边，答案比较简单。139团队，就是1个项目经理，带3个高手（师傅），每个高手再带3个新手（徒弟）；师傅全权负责自己及3个徒弟的工作，从进度，设计到质量；师傅会选择关键时间点与徒弟碰头，以便用最少时间解决最大的问题（这种活动称之为“松结对编程”，在我的“敏捷开发松结对编程系列”中有详细描述）；师傅对徒弟的代码进行代码审查；平时徒弟有问题会师傅。在这样一个团队中，项目经理收入最高，

师傅次之，徒弟最低。薪水相对固定，如果位置没有变化，无论自己一个人多编码了还是帮别人太多导致少编码了，都差不多。因此要想拿到高薪，不是在短期内多编写几行代码或关闭几个Bug，而是看长期：新人先要达到独立工作能力，然后帮助团队扩张也就是自己带徒弟成为师傅，最后成为可以独自担当一种业务的人也就是项目经理。在这种团队中，人的收入增长不是通过内部竞争，而是通过内部协作做到的。

因此它是一种兼容敏捷团队精神的个体绩效考核方法。

二、外部度量

正如前面所说，外部度量是为了让团队对外透明而做的度量。有这样几个原则：只做外部看得懂的度量；只做外部有人看的度量；尽量少做度量。

典型的外部度量是给PO也就是产品经理所作的用户故事的完成情况的度量，比如白板上处于完成状态的故事的数量。这里说故事而非任务，是因为对产品经理而言，任务完成没有意义。

生产率也经常是一个被度量的内容，但现在尚无好的方法。有些团队经常使用“故事点”来做度量，方法是：先挑选一些以往的故事作为典型故事；为每个典型故事分配一个点数（一般是当时完成此故事的天数）；每次估算新故事时查先看更像哪个典型故事，然后再根据难度差别为新故事设定一个点数；实际完成后，点数/天数=生产率，也就是实际完成点数的速度。这个方法看似有效，但因为大家对典型故事的熟悉程度差别很大，典型故事也很难选择，导致实际使用的人很少。另外一个问题是不同项目的典型故事和点数是不通用的，因此无法做比较。好不容易做了一个想对外发布的生产率，却无法横向比较只能自娱自乐，显然存在问题。未来，这个问题可能会被功能点生产率解决，但现在还没有看到有人尝试。我们自己做了一些宏观尝试，但没有每个迭代都做。

此外还有一些度量，但取决于实际的需要。比如在线运行的网络游戏，会度量迭代交付周期长度（直接影响响应时间，因此一般越短越好）。这些度量不全是明确量化的，但公司上下都能说出一个大致的数字来，因此也算是度量。

三、业务运营与收入度量

既然敏捷开发说是“交付客户价值”而非文档或代码的，那么一定能改善营收或其他运营指标。

比如市场占有率，每日点击数，平均在线人数，每活跃用户收入……等等，这些营收应该都是做敏捷度量的人所需要关注的。Scrum设置了PO来收集、描述、排序最重要的需求，又安排团队在自组织下高速生产出来，通过评审会后交付给客户。因此如果这一切都是真的，没有理由在实现敏捷开发后业务营收不变或下滑。当然可能很多因素制约了企业的营收情况，但如果既不度量也不关心，直接来一句“由于情况很复杂，因此企业营收与研发没有直接关系”，显然只能被作为自我否定和消极逃避的借口。

现在有些团队如网络游戏和搜索引擎公司，已经把这些数据引入了开发团队的度量中。但一般不做绩效考核，而是让大家知道最近我们的研发在市场上的实际反馈如何。

四、内部创业

任何时候想度量的时候都要问：为什么要度量？答案是：度量的目的是为了改善目标。

那么如果说企业的终极目标是盈利，而却有一种方法不需要度量就能提升盈利，那么应该怎样？

当然就不用度量了！目标永远比方法要重要。

比如现在网络游戏团队普遍有一个政策：20%的营收归项目作为奖金。一般这个数字从几十万到上亿的规模，而团队往往不到100人。他们不在通过繁杂的绩效考核系统中的度量项发放奖金，而直接把收入变成开发人员的动力。这种方法跳过了所有繁琐的度量过程，直击问题实质，是一种不度量而改善目标的典范，可以理解为敏捷开发“可运行软件超过繁琐文档”这种尝试弱化中间过程的思想的产物。

当然，即使不能“内部创业”，也一样能做到类似的效果。比如降低程序员的基本工资，代之以把公司产品销售的一部分钱拿来做奖金。程序员自然就开始关心客户需求，尝试理解客户到底要什么不要要什么，尝试自觉改善质量防止客户流失，尝试自己带徒弟以便让团队成长进而增加产品竞争力……很多效果是通过度量都难以进行改善的。

最后这个问题的答案，其实回到了问题的本质：敏捷开发提出的目的，其实还不是敏捷宣言中的那四句话左侧的部分（个体与交互，可运行软件，客户协作，响应变化），而是通过这四点来帮助企业提高营收。因此，敏捷开发的度量，换言之敏捷开发要改善的地方，不能只是一线工程实践，而应该是“客户价值”的改善，也就是营收的增加。否则敏捷开发就可能长期处于基层游击队活动，在敏捷热潮过去之后，被证明又是一个重过程不重结果的空架子而已。

记者：您是否能谈谈在项目中，敏捷测试人员的职责和技能要求？

陈勇：关于这个问题，有趣的是：“测试活动”比“测试人员”在敏捷开发中出现的频率更高。原因就是敏捷开发中提倡跨职能，也就是所有人都对开发、测试负责，从测试活动的目标的转变上就能看出来。

传统认为，测试人员是帮开发人员找Bug的，这是个误区。其实，开发人员才是负责找Bug的，尤其是自己找自己的Bug。打个比方，如果我们在自己家里忘了钥匙放在哪里了，一定不会把家里翻个底朝天，而是会尝试找自己以前习惯放钥匙的地方；但如果让一个客人来找，就不那么容易了，因为客人不熟悉家庭环境，也不熟悉我们放钥匙的习惯。家庭环境就是软件，而钥匙就是Bug，测试人员则是一个不太熟悉家庭环境的客人，开发人员才是那个放钥匙的人，尽管他暂时“忘了放在哪里了”。而且长期而言，只有开发人员才能改变自己的习惯，比如在墙上钉个钉子，习惯性地把钥匙挂在上面。我们在一个项目中密集实践过代码审查活动，在短短27天时间里，通过每天花费半小时观察、记录、分析自己的缺陷，个体缺陷率降低了一半之多。而且好的习惯和规范一旦固化下来，以后无需花费额外时间也能在一定程度上保证代码质量，可谓一劳永逸。

找bug的活被抢走了，传统测试人员怎么办？有两种出路，或者说“职业生涯规划”，都比原来传统测试人员更有前景。

一、有开发经验的测试人员应该转向“开发测试”

“开发测试”是游戏领域的一种工种，对应的是完全不需要对开发有了解的“黑盒测试”，我们可以把它的定义扩展一下用在这里。开发测试在开发活动中主要负责自动化测试、持续集成、回归测试等用于快速保证产品质量的活动。

为什么叫“快速保证产品质量”而不是“发现Bug”呢？因为开发测试程序的人其实不应该是这

里的开发测试，而是开发某个功能的开发人员本身。正如之前所说，具体的功能开发者更清楚功能是什么，可能有哪些潜在的问题，如何验证最好……所以开发并第一次执行（为发现缺陷而执行）的是开发人员；但如何让众多测试用例自动地全部或有选择地再次运行，如何保证更高的运行效率、如何准备自动集成环境，则是开发测试人员的工作。

如果有可能，个人感觉最好不要设置全职的开发测试，而是视实际情况由不同的开发人员兼任。这样更容易平衡工作量、保证开发与测试的衔接有效性。

整体上这个出路偏向开发人员多一些。

二、“没有开发经验”的测试人员应该转向“产品经理”或“专业测试人员”产品经理不说了，因为出路很窄。这里说说专业测试人员。

有很多时候我们觉得原来的测试人员不就是“专业测试人员”吗？其实不然。很多测试人员不过是看看说明书，点点鼠标，填填Bug而已，很难说上“专业“二字。

个人感觉对测试人员而言，“专业”有两重含义。第一，对所在领域的质量问题和解决方法有深入理解。比如特定的软件，可用率是首位的（比如网站，允许瘫痪但不能总瘫痪）还是故障率是首位的（比如飞机，不允许故障）。Windows经常死机，但是恢复速度很快，用起来也很简单，这在家用软件中就是可接受的“高质量”标准；但用作服务器，可能就有点问题。如果不理解这些就开始测试，结果肯定是事倍功半。第二，对测试方法论应该有深入理解。软件仿真，持续集成，灰度发布……这些都可以称为测试方法论或质量方法论。“专业测试人员”用这些方法论指导前面说的“开发测试人员”进行测试，是一种很好的工作方式。

我曾经在松结对编程与139团队系列博客中多次提到一个23个开发人员加2个测试人员的团队，其中两个测试人员就是专业测试人员。尤其是其中的测试经理，会定期基于业务的需要向我们提出测试要求，而我们则帮她编码实现。这种搭配方式最终促成了产品的高质量，它现在正运行在国内60%以上的数字电视台。

这里的“没有开发经验”被打上了引号，是因为我无数次听到有人说“我没有开发经验，也不懂开发”，但这个人可能是与开发人员共事很多年的资深测试人员。其实，与相对论相比，开发更像是外语，就是外国乞丐学多了也能学会的东西。因此实在不能用技术壁垒作为借口，要做好可能很难，但要了解的确不难。要做好测试这个工作，不去侧面了解和理解开发，是很不现实的。

记者： 在敏捷实践的过程中是否一定要使用一些敏捷工具？能否推荐一些您认为比较好的工具？

陈勇：因为我自己也在开发一个敏捷工具“火星人”，所以就不直接推荐工具了，呵呵。不过下面可以介绍一下一些一般原则：

一、一定要选择适合自己行业领域的产品

有些产品可能是为大型制造业、银行、电信等开发设计的，那么尽管也叫“敏捷开发”产品，但里边会带有很多大型产品设计中所需的元素，比如UML，Portfolio Management（可以理解为大团队或多团队的管理）等内容。这并非像有人说的他们在“挂羊头卖狗肉”，而是在特殊行业实施敏捷所必需的适应。反之，简单的电子白板工具往往反而不适合在这些行业应用。

通过翻阅供应商提供的客户清单可以有所了解，而产品的主要功能和主张的方法论也能提供一些线索。这些资料往往印刷在宣传资料或发布在网站上，公开且不易被轻易修改。

二、根据“数据持久性”选择产品

所谓数据持久性，就是指数据要保存多久。如果是电子白板类产品，产品选型可以稍微“随意”一些，因为电子白板上的东西多数零散、颗粒度较小，很少需要几个月之后还要查看。所以可以大胆尝试这些产品。其他比如计划管理的工具也比较容易更换，毕竟完成了的项目的计划被访问的次数和必要性会迅速下降。

而对于需求管理、缺陷管理等产品，尤其是前者，则需要慎重选型。视行业的不同，这些产品可能需要考虑到未来3～5年乃至更长的数据保存问题，所以需要慎重一些。另外这些产品往往跨多个职能部门，选型时需要多方同时协调。

三、未必一个公司只用一个产品

All In One是每个公司的梦想，不过却不现实。我培训或咨询过的一些大型电信企业，其涉猎的领域之广往往令人惊讶。有家公司在做计费系统、BOSS系统（业务和运营系统）的同时，还在做移动互联网软件甚至游戏。前两者对需求遵从性、成本控制的控制是核心，因为它们往往涉及甲乙方合同；而后者则对创新性要求更高，喜欢“拥抱变化”。

这种情况下管理层应该深刻理解不同领域的管理要点。比如甲乙方项目中，“谁在做什么”这类任务细节往往是重要的，因他它能保证所有人都有活干，防止空置资源的浪费；对完成功能的多少的掌握也是重要的（最好使用功能点），因为它能表征进度。但在移动互联网软件中，“现在我们在做什么”“什么时候这个功能能做出来”“这个功能做出来反馈如何”才是重点。

由此可见，领域要求不同，方法论就会不同，管理工具自然也可能不同。除了一些简单的白板软件“都能用”之外，复杂产品很少能适应如此巨大的差异，甚至说也不应该适应所有领域使用。

在这种复杂情况下进行选型的时候，领导层或领域专家应该给出自己对管理的核心需求，然后再放手让团队进行自由选择，只要验证核心需求被满足就可以了。

袁斌：敏捷开发在项目中的解决方案

■ 敏捷开发方法的流行正是因为大家发现了传统瀑布方式无法适应从改变，革新而驱动企业的动态管理。如何在项目开发中能持续满足不断变化的用户需求让越来越多的开发和管理者受到了重视。

敏捷开发方法的流行正是因为大家发现了传统瀑布方式无法适应从改变，革新而驱动企业的动态管理。如何在项目开发中能持续满足不断变化的用户需求让越来越多的开发和管理者受到了重视。敏捷开发成为了众多团队的制胜之道。下面，我们专访了北京迅思威尔科技有限公司的资深敏捷教练袁斌老师给网友们分享了敏捷在项目实践中的一些经验。

袁斌，工学硕士、MBA， Scrum、AUP、Agile modeling、XP、kanban等资深实践者，资深敏捷教练。近20年中就职于全球性公司从事软件和产品的开发。曾任Anoto产品和Mino产品中国区软件开发总监，利用Agile & Lean实现产品快速交付，应对变化的需求。超过8年敏捷开发&精益开发的实践过程中，在电信行业、外包团队、互联网产品等多个行业积累了丰富的经验。

以下是视频采访文字实录：

记者：51CTO的网友大家好，今天我们请到的嘉宾是北京迅思威尔科技有限公司的资深敏捷教练袁斌老师。跟我们讲解敏捷开发项目实践中一些主要问题，以及对未来几年敏捷发展这一块的一些看法进行分享。首先请袁老师跟网友们做一下自我介绍。

袁斌：大家好，我是袁斌，来自迅思威尔，我自己有超过八年的敏捷的实践经验，所以我对自己的定位是一个资深的敏捷实践者。OK，谢谢。

记者：那么在敏捷当中，很多人都看到了它的成本控制，它能够降低项目开发中的一些浪费。但是往往适得其反，不但没有降低，反而增加了开发成本。在开发成本控制这一块，你怎么去理解？

袁斌：其实对于成本来言，每一种新的方法一定会有它的学习曲线和学习成本，往往在开始的时候，它的结果会比现有的情况还要糟糕，它会有一个U形的曲线。所以这里面更多的是怎么样降低你的学习成本和学习曲线。如果在最开始使用敏捷的时候全盘采用整个敏捷开发实践集合，较高的学习成本和学习曲线会使实施敏捷的效果大打折扣。所以我这里的建议是，对成本控制而言，首先找你在研发过程中间最大的痛点，在痛点的基础上，然后在敏捷开发的众多实践里面去找针对这个痛点的实践集合。这个实践集合的原则就是学习成本要低，而且它能够持续改进，对你的痛点起的效用最好。这样的话，不断地改进，持续控制成本，就可以获得让人满意的投资回报率。

记者：在敏捷实践过程当中，是否使用到一些敏捷工具，能否推荐一些比较好的工具？

袁斌：敏捷工具是这样考量的：看你产品的周期，如果你产品周期相对比较短，比如说你两三个月，那么我个人并不建议用敏捷工具来考量、管理你的敏捷开发的过程。如果是周期比较长的话，我是建议用敏捷管理工具。因为原因主要在于，第一，你要去积累很多数据，这些数据会有利于分析你的过程中间，哪些是好的？哪些是不好的？，然后有针对性的改进；第二，对于整体的需求管理，包括你一些技术方案的管理，需要一个工具能把它整体管理起来。

记者：在项目当中，敏捷测试人员的职责和一些技能的要求是什么样的？

袁斌：我觉得在敏捷的开发过程中间，测试人员更多的是能够帮助团队，帮助团队最后的交付

能够有一个保证。所以说敏捷的测试人员首先应该是尽早测试，更早地在测试周期的前面进入。比如说在一个迭代开始的时候，在计划会议他就能够介入，他给出很多从用户层面对需求的看法，能够帮助团队更好的了解需求。同时通过把握测试的风险，尽早的给研发团队“bug出现风险高”以及“用户使用频率高”两个层面的用例，帮助团队控制研发过程中间的风险。

我觉得还有一个很重要的是持续地测试，能够去帮助团队，特别是研发团队，不但是说我能够尽早，而且不断地持续地进行这些测试，及时的把风险反馈给研发团队。

记者：据我了解，很多朋友说你在测试这一块，好像是他们不需要专门的测试人员，你觉得这样的公司对于开发一个项目的时候，有没有什么影响?

袁斌：有一些公司会有，比如说像Facebook，它会说我没有专职的测试人员，它是工程师文化。但我个人认为在我看到的很多项目里面，测试人员是必要的。我说一下我的观点，我觉得如果没有测试人员对工程师的要求非常高，因为他要承担专业测试人员的技能。

记者：对他个人的一些技能，什么要求都会高。

袁斌：对，我觉得术业有专攻，测试人员更多的从用户的场景对产品进行测试，特别在非功能性需求的测试，这方面的专业技能要求也是很高的。

记者：做到敏捷开发，每个团队都要经历一个转型期，在转型期还需要每个团队根据自身的不同找出合理有效的解决方法，一般是如何处理这个问题?

袁斌：团队在做转型的时候，我们看到的很多团队，包括我们自己的客户，最大的问题在于，敏捷开发有很多的实践，如果把所有的实践都拿过来。然后全部用在团队中间，但是每个实践都有学习成本。所以说都拿过来以后，团队很抵触，而且也不能解决团队的实际问题。因为太多了实践以后，大家觉得很疲惫，有时候适得其反。我个人的建议是这样子，在转型期的时候，用痛点驱动的方法。所谓痛点驱动就是，找到团队目前一个很严重的痛点，在敏捷实践中间，找一些实践集合，不要全都找，是找一些实践集合，去把痛点解决掉。然后再看下一个迭代的时候，继续找目前最大的痛点，然后再用一些实践集合把它解决掉。你不断地这样去做的时候，团队抵触很小，他觉得OK，把我的实际问题解决掉了，他会很愿意采纳敏捷。

记者：您刚才说痛点驱动，能否解释一下?

袁斌：所谓痛点驱动，痛点其实就是最大的问题，是我在研发过程中间碰到的最大问题。比如说我可能在这个过程中间，我最大的问题是迭代交付延期。

记者：延期?

袁斌：对，我延期了，也有可能最大的问题是对需求了解很差，或者说人员流动是一个最大的问题。那么比如说人员流动是一个很大的问题，这个时候敏捷实践里面，你不见得非要用测试驱动开发，要用用户故事描述需求，可能是结对编程，可能是团队的代码负责会更有效。也可以强调计划会议里面整个团队都要参加讨论，减少学习债务。这些实践可能是对你当下的痛点会有更大的帮助，建议首先采用这些实践，我的观点是

这样的。

记者：敏捷过程当中，我认为最重要的是一个需求、拆分与分工，你能谈谈如何进行它需求合理地拆分，以及分工吗?

袁斌：我的观点是，如果是需求的话，尽量采用一种拉的方式，所谓拉的方式是说，我们研发的流程就是从需求分析、再到设计、编码测试、然后交付客户。我的观点是，我们应该是从客户这一端开始，客户这边认为，他最需要的是哪些需求，然后对这些需求做比较详细地拆分。所以这样在Scrum里面，它有一个叫做Product Backlog，翻译过来叫需求列表。

这个需求列表里面高优先级，一定是客户最想要的，所以需求拆分我建议从高优先级的，对用户价值最大的需求拆分会比较好一点。

记者：在敏捷开发过程当中，Scrum和XP，也就是说我们说的直线编程，在国内据我了解，使用Scrum占了大部分。

袁斌：没错。

记者：你能介绍一下Scrum和其他几件方法最大区别在哪里？如果是Scrum，它的优势在哪里?

袁斌：其实Scrum最主要的特点在于说，它是个轻量级的框架，通过短周期的迭代，交付最有价值的产品，可以从容应对变化的需求。它最大优点我认为是，它没有介绍很多的实现细节，具体要做到怎么样去交付一个很有价值的东西，它没有想洗介绍。我认为这可能是它最大的缺点，也是最大的优点。它可以包容很多其他的方法，比如说像极限编程（XP）。它可以把XP里面很多的工程实践融进来，帮助它提高交付质量，而且它可以包容像精益软件开发，消除研发过程中的浪费。所以Scrum，我认为它最大的优点，是个轻量级的框架，易于包容。而且从另一个方面说，在国内能够很好地流行得益于它是一个管理型的方法。其实Scrum是偏向于敏捷的项目管理，项目管理在国内很容易获得认可。所以我觉得它在国内能推的比较好，这两点是比较重要的。

记者：极限编程比较注重细节方面的一些问题，可以这么说吗?

袁斌：极限编程中的工程实践很有借鉴性，但是对于工程师本身的要求相对比较高。Scrum相对而言对工程师的要求没有那么高。

记者：就是说一般我们国内一些创业型的公司，它的项目都可以用Scrum的方式。

袁斌：一般而言是可行的。同时Scrum不但用在软件产品的开发，还有很多其它的行业都可以用Scrum。

记者：就是说不仅仅限制在软件开发这个行业里面?

袁斌：没错，是这样的。

记者：那么如果能得到准确的数据支持，敏捷实施能够更好地发展下去，敏捷实践下的员工，程序员的工作指标如何衡量?

袁斌：我明白你的意思。我们的实践中间是这样子，把程序员的工作指标分为两个部分，一个部分是团队，我们首先不关心每一个人工作量多大，我们想要关心这个团队的每次交付，是不是可以接受？这样大家会有一个团队的概念，整体的概念。另一个部分，我们需要关注个人，关心个人的时候，一般会关注四个层面，第一个是工

作质量；第二个是工作量；第三个是主动性，因为敏捷开发里面，短周期的迭代，对风险地控制要非常高，要求主动沟通的意愿要很强，有助于风险前移；第四个我们认为是帮助，就是帮助团队。

记者：当一个敏捷团队工作时，有时候他的项目流程会暴露出来，也就是我们常说的项目透明化，这是不是可以称为敏捷开发流程的一个过失，或者是说你对于项目透明化有什么见解？

袁斌：首先项目的透明化在敏捷开发里面，他要求的非常高，在我们的实践中间会发现，如果你把这个项目透明了，你会减少很多沟通的成本和障碍，现在我们能看到的项目透明化方式更多的是白板。

无论是物理白板还是电子白板，大家把这个项目放在透明的环境里。很多的团队会做一个项目墙、任务墙，把所有的状态在墙上布置下来。还有一种方式就是电子白板，电子白板有很多的工具。

这两种方式都可以把你的项目风险暴露出来，项目透明化以后，其感兴趣的项目干系人都会来帮助我们这个团队，发现项目可能潜在的风险，同时会帮你解决掉。

记者：对，在Scrum会议的实践实际上是实践控制当中最重要的一件事件。

袁斌：没错。

记者：这更是团队做改进最佳的时机，那您认为Scrum会议在团队当中如何建立起来？在Scrum这块非常看重它有一个回顾会议，这个如何进行？

袁斌：回顾会议其实在Scrum里面起到了持续改进的最用，敏捷非常强调持续改进。

回顾会议在Scrum里面是持续改进的一个非常好的机会。但是现在能看到的是，回顾会议有很多时候团队流于形式了，觉得开的没有必要。更多的情况下我认为是，这时候回顾会议没有起到应有的作用。我认为一个回顾会议开的比较好首先是数据驱动，所谓数据驱动就是，我们通过数据去得到这个迭代里面哪些是最大的问题。找到问题以后，团队去分析它的根本原因，再去找到它的解决方案。然后把两三个好的解决方案，在下一个迭代里面去把它固化。回顾会议最核心的是根本问题的发现，所以我个人很推荐使用数据的方式。比如说Bug分类，计划和实际的偏差。比如说你每天八个小时的工作时间，你是三个小时有效工作，其他的五个小时我们可能会分析浪费在哪里？这些点都会把问题暴露出来。

记者：在Scrum会议中，除了回顾会议以外，还有没有其他的一些会议？

袁斌：就是Scrum里面？

记者：对，Scrum里面。

袁斌：Scrum里面其实有几个比较重要的会议，第一就是它的计划会议。在一个迭代开始的时候进行。计划会议是怎么样保证在迭代过程中间能够把所承诺的东西高质量的交付。第二个推荐的会议是每天早上的一个站立会议，15分钟。这个站立会议可以和我们前面你提到的白板一起使用，做到项目透明化。站立会议如果想开的好，最好所有成员在白板前面来开。这时候每一个成员都会把各自的工作状态和风险在白板上暴露出来。

还有一个会议是评审会，在迭代要结束的时候，我们来看这个迭代的交付是否能够满足迭代最开始的目标。然后就是回顾会议，回顾会议是最后我们要看哪些做的不好？哪些做的好？在下个阶段如何改进。

记者：在目前国内出现很多敏捷教练这个角色，您是一位资深敏捷教练，你认为敏捷教练在一个团队中，他主要的工作有哪些？而且敏捷教练在Scrum工作中建立起一个角色的转换。

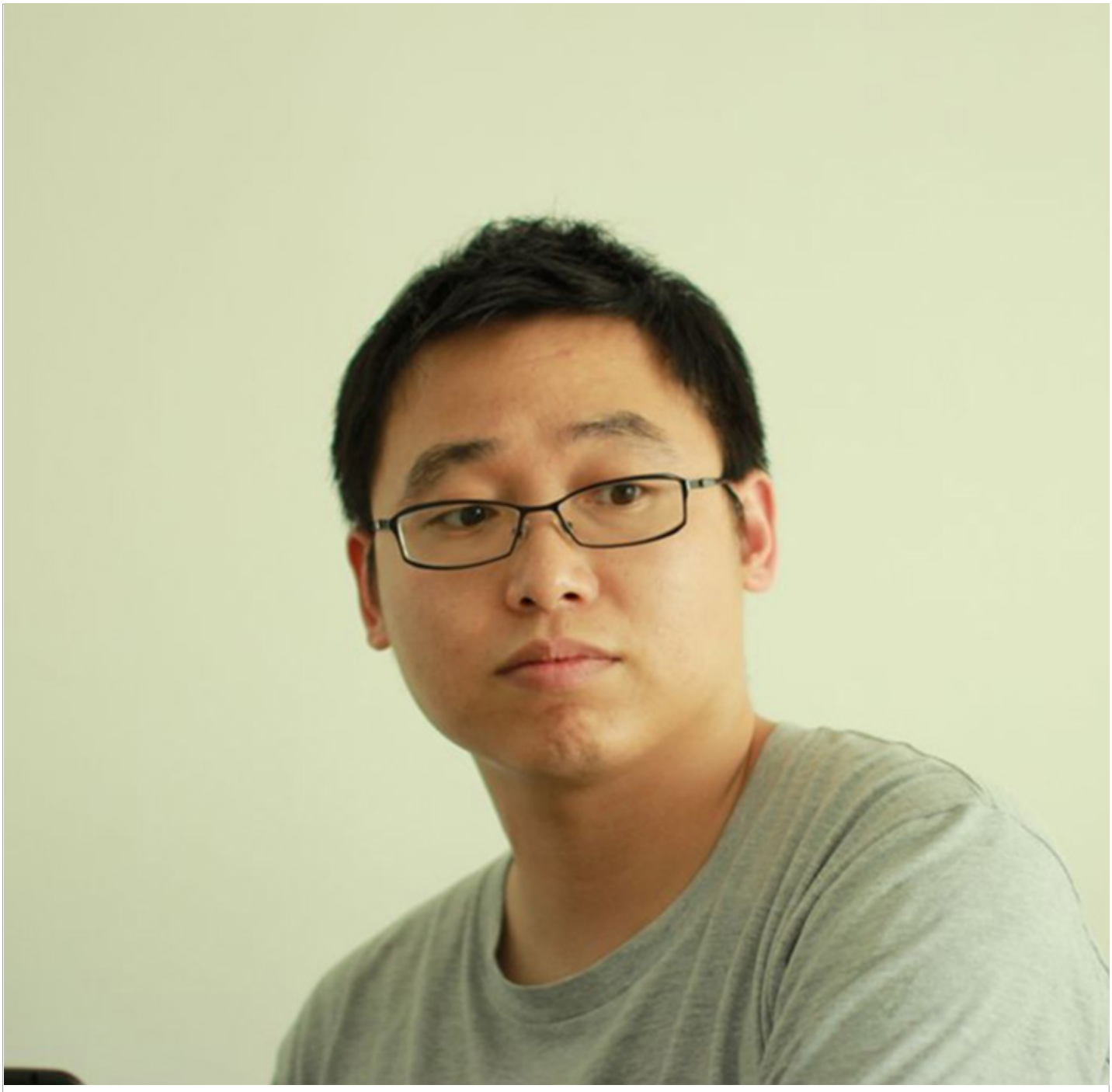
袁斌：我觉得是这样，一个教练在一个团队里面，他最关键的是随时能够帮助敏捷在团队里面能够落地，包括敏捷的思想，敏捷的实践，能够让它落地。如果再细里说的话，它是能够帮助团队，通过敏捷的思想和实践解决团队现有的问题，这个也是我自己实际中的做法。也就是说，敏捷教练和团队一起来工作，通过敏捷的一些实践，也包括其它开发模式的优秀开发实践，来帮助团队解决问题。这时我认为敏捷教练最应该做的。

记者：在项目中他是一个不断持续改进地过程，不能按部就班地说，敏捷是怎么样就怎么样去要求他们这么做。

袁斌：对，因为很多敏捷实践背后是有假设的，它假设团队成员的技能达到什么程度，假设公司的文化会如何支持尊重、平等，但是这样的假设如果在项目中并不存在，如果我们把这些实践照搬过来，一定会有很多的冲突。

记者：在对未来几年敏捷开发的发展希望看到哪一些方向？

袁斌：我觉得未来几年，我的希望是，敏捷可以来落地，它可以真正帮助团队解决实际问题，提高他的质量，提高他的效率，提高他交付的商业价值，这是希望看到的。第二个就是说，我非常希望大家忽略敏捷这个词，不再是说，我用敏捷了，或者说我没用敏捷，而是说把敏捷这些实践和它的思想，甚至于说，我们基于它的思想，同时基于我们这个团队的现状，做一些定制化，形成我们自己企业和团队的一套方法。然后我把它落地了，给团队带来实际的好处，我们不再谈是否敏捷，而是实实在在地在持续改进。



在国外软件企业中，几乎将近半企业是已采用敏捷方法进行开发，随着近年来受软件外包和外企的带动，敏捷开发在中国也出现了日渐普及的态势，如腾讯，IBM等等内部几乎所有的开发团队都在实施敏捷方法。

专访程显峰：敏捷在团队中的实践与发展现状

无论是在软件或者管理行业敏捷开发已成为众多高效开发团队的制胜之道。敏捷开发可以说是在开发中面临迅速变化需求中的一种快速开发能力。当然，敏捷不仅是简单的快速，而是在开发过程中短周期的不断改进、提高和调整；当然也不仅仅是一个版本只做几个功能，而主要的是突出重点。

在国外软件企业中，几乎将近半的企业是已采用敏捷方法进行开发，随着近年来受软件外包和外企的带动，敏捷开发在中国也出现了日渐普及的趋势，如腾讯，IBM等等内部几乎所有的开发团队都在实施敏捷方法。敏捷开发的流行绝非偶然，因此，51CTO记者采访了AdMaster（精硕科技）首席布道师程显峰，讲解了敏捷开发在团队中的实践以及发展现状。

简介：程显峰，毕业于悉尼大学，《MongoDB权威指南》译者，MongoDB中文社区创始人。Emacs使用者，Ruby写手，Scheme爱好者。AdMaster首席布道师，负责团队建设，人员培训，新技术普及，还有一些公司技术PR的工作。

以下是相关的采访实录：

记者：有些人认为敏捷开发并不适用于水平一般的程序员或团队，您是怎么认为的?

程显峰：至于敏捷开发是不是用这种程序团队，我认为特别适合训练有素的团队。反过来说，如果不是一个训练有素的团队，实施传统软件开发也会失败，所以我觉得基本的要求是跟原来的一些要求基本一致的，沟通能力、协调能力，对于项目变动的控制能力都是一样的。包括传统软件开发当中讲的项目可控，整个项目可复制，同样的资源配比后还可以复制这个项目，如果原来的这些做得比较好的话实施敏捷项目就相对容易一些。

记者：很多人为了编写易变更的代码，在采用敏捷时，抛弃许多习惯用法，由你的经验出发，如何去看待这个问题?

程显峰：从我在实施的过程中第一点需要强调的就是大部分跟传统的软件工程还是一样的，现在异化的部分过于被强调了，搞得大家以为是完全新的、不一样的东西，传统软件包括需求管理、进入控制、质量管理、测试体系，这些东西在敏捷里也都要体现、都要实施，而且要求可能还会更高，并不是把一些传统的东西抛弃掉了，既没有进入管理又没有版本控制，几个人一商量就定了敏捷，这是完全的一种误解，而且还是要好的工程基础才能实施。

记者：所以传统的方法对于程序员的要求还是比较高的吧？敏捷这样的功能经过修改之后就可以推出一个小的版本?

程显峰：并不是说推出小版本就降低难度，这是没有因果关系的。比如说这小的版本即使推送上去，如何保证这个小的版本是对的？你也需要大版本的方法，比如软件测试、进度控制、需求管理，这些东西都得有，你才能控制好。虽然你推得比较快，看上去可能是一个一个的，如果你要是推的没有章法的话一样也是乱套。不是你想要的东西，也会造成大量的浪费，我是这么觉得的。

记者：对于一个失败的软件项目来说，您认为敏捷方法是它的救星吗?

程显峰：对于一个失败的项目就要说出失败的原因，这才是解决失败项目的办法，敏捷的方法不是，这个逻辑就是有问题的，我觉得这个没什么好说的。

其实我觉得敏捷的方法不是任何失败项目的救星，这是肯定的。至于失败这个事情，看上去可能是分工的原因，有可能深层次的来讲是不交流、不沟通的原因。敏捷比较注重强调沟通互动这些方面，它是有强调这些东西，如果你的问题恰巧是由于不沟通导致的，可能采用了敏捷的方法可能会有极大的改善，但也非常取决于实施的人和分析的效果。好多人说敏捷就是一个框，什么东西都可以往里面扔，实际上也确实有点这个样子。

我觉得敏捷的方法更多的是注重发现问题的框架，就是能让你更快地发现问题、暴露问题。至于怎么解决问题，原来能解决的就能解决，要是原来解决不掉，敏捷也帮不了你。但是它能帮助你发现问题，比如说原来你有沟通问题，但是很长时间才能暴露出来，这个东西能让你在短时间内暴露出来，那么就有助于解决这个问题。

记者：想要做到敏捷开发，每个团队都要经历这样一个转型期，那么在转型期还需要每个团队根据自身的不同，找出合理有效的解决方法。你们团队的是如何来解决的?

程显峰：我是倾向于比较温和地对团队进行改进，实施的东西尽量不要让团队有一个明显的转型期，比如版本控制，原来做的版本控制、提交力度、上线流程这些，我会给大家介绍一个新的系统，但是并不会太影响大家的工作，适应起来又很快，这样的改进效果是比较立竿见影的，而且影响又非常小。这种改进不是偶尔一次才会用到，而是天天都能用到，比如说调试方法，每天能省一分钟就行，我特别喜欢这样的改进。

大家可能都觉得敏捷特别神奇，每天能省百分之二十的时间，现在我还没发现这种现象，我特别不相信这种东西，就是不想省百分之二十的时间就不会让大家经历一个很长的转型期。比如你告诉大家一个技巧，每天能省两分钟，这种东西大家天天用实际积累下来的效果才是比较明显。你要是教给大家一个方法，比如能省二十分钟，实际上这种方法看上去是非常漂亮，但它不是什么场景都能用得到，所以我就不太喜欢这样的方法。现在大家吹嘘的敏捷方法可能有一部分也不敏捷，我在这个团队里也不太愿意教那部分，比如说每天早上例会的时候站在那里说一通。我也跟他们讲过，他们有的话可以，没有也OK，有些东西我的要求就比较死，比如上线的流程、可控性，我在这个方面的要求就比较死，开不开会对于我的项目没有什么影响，但是上线的流程就不一样，我要往回退的话就要花成本。

对于小的团队来讲真的是可有可无的，因为团队的人都坐在一起工作，未必需要一个站到板子上的形式。现在有人把形式看得很重，本来例会是很简单的，五分钟就开完了，但是吵起来了，开不完，因为一般都是开早会，吵完以后心情就不好，所以效率就下来了，这也是很正常，但是你不要想着漂亮的东西，我对漂亮的东西感觉都不实用。

记者：会不会因为不吵这个架把错误带到真正开发的过程中造成成本的耗费?

程显峰：会，所以我们要在实际的工作当中发挥，其实例会也不是吵架的会，还有很多的会可以发现这个问题，比如通过Review，可以有各种

各样的方法去发现，不用拘泥于这种形式上的东西。其实最难改变的往往是那些小的习惯，比如他就愿意用这个手指头去敲键盘，但是这可能会影响他的效率，他不愿意改。这种习惯的话改的阻力反而比较小，你让他去开个例会他觉得耽误时间，但是要让他改这个的话他可能会改。你想他要敲多少？天天可能都要去敲，这种感觉可能是持续的。

记者：这种其实是属于比较个性化的调整，每个人的使用习惯不一样，有些人可能喜欢用这个手按着，有些人喜欢用那个手按着，你要主动发现各个人在工作当中需要调整的地方？

程显峰：我一直认为团队就是一个泯灭个性的过程，而且这一点我是非常讲究的，包括编辑器的使用习惯和配色都有强烈要求，不像别的看上去那么自由，坐在哪里无所谓，但是你用电脑的方式必须是我规定好的。没有一个共同基础的团队根本就不叫团队，总有一些东西是有共同基础的，而且一些高效的团队很可能所有的东西都是共同基础，就像特种兵出去打仗，一个手势就得去杀人，你还能再问这个是什么意思？就像我们用别人的电脑，或者在团队当中用其他人的电脑，那些配色习惯都是一样的，我们沟通交流起来就会非常容易。

现在新人如果不配色，直接就卡掉，新人培训就不合格，所以必须要有配色，包括快捷键的使用，不能人家给你一套代码你还慢吞吞的，这会非常影响别人的心情。因为有些忽悠师天天都去忽悠各种敏捷方法，但又不注重工程实践，不注重观察细节，所以就做不到。他们不会注重这种实践，比如咱俩是木匠，我天天使完的刀子随手一扔，天天你给我收拾，虽然我是大牌，但是你也不愿意。你和我都收拾的很好，大家都用一个工具箱，这样才能省下很多时间。我就觉得能省时间的话，就是在任何层面上都是节约和避免浪费的表现。

记者：如能得到准确的数据支持，敏捷实施能够更好地开展下去，那么敏捷实践下的程序员工作指标将如何衡量？

我是觉得这个数据比较难以衡量的，因为衡量有意义的必须是在某种固定的场景下。比如对于我这里来讲，我可能会要求程序员开发一个标准模块的实践，但是对于别人来讲，这个他可能不太注意。从实施的角度来讲，我对怎么度量这个事情想的不是很多，我是比较相信主观看法的，因为我在实施的过程中是跟他们在一起工作，不需要用那些数据来告诉我他们已经进步了，他们做外部咨询的就比较注意这个事情，因为他们需要给那些老板写报告，我不太需要这个东西，我对其它的产品负责就可以了。

记者：刚才提到你特别关注的除了工作规范和开发规范上的细节，还有就是发布流程。

程显峰：是这样的，发布流程实际上就像互联网团队的一个核心，就是有一个标志，如果我拿到一个需求，最后到发布有多长时间，这是看整个团队效率的标志性衡量，现在很多人都讲敏捷，其实软件度量是一门学问，究竟度量什么大家非常有争议，但是我感觉没有太多争议。所有的度量尺度都用时间，因为时间是比较公正的，没有差别的。比如你用代码行数，大家都说代码行数不是一个标准尺度，你用C写的五百行和用Java写的五百行到底能不能衡量？但是你用时间的话，我这里一秒和在美国那里一秒是不是都一样的？所以所有的东西都应该用时间来衡量，很多

东西都可以转化成时间。比如上线效率就可以转化成时间，拿到一个需求到最后推上线就是一个标准的时间，你能在多短的时间之内完成这个事情，这是你团队效率的一个提升。

还有一个非常重要的提升，就是当你发现线上的东西有问题，多长时间能回退到上一个版本？我们可以在秒级之内回去，就是一分钟之内退回去。这是整个团队开发效率的一个显著的管理，如果你做不到这一点的话，比如你已经发现Bug了，第二天才能把这个更正，其中的时间全部都浪费了，而且对于一个线上系统就是损失。我是讲比较实在的东西，对于开发流畅要把控的特别严，怎么才能把控上线的这些东西呢？就是要把上线的过程和开发的过程、测试的过程全都紧密地联系在一起，这是一个整体，而不是把它们割裂。包括很多传统企业这个方面做得也非常好，并不是敏捷独有的。所有的东西我坚信都不是敏捷独有的，对于优秀的工程实践来讲都是通用的，整个过程都是非常好、非常快。

记者：在敏捷方法流程中具体 实践的有极限编程（XP）和Scrum等等，Scrum目前团队用的是比较多的，对于极限编程不知道您有没有了解，据我了解很多团队用过Scrum的都不喜欢极限编程的实践方式，能说说您的观点吗？

程显峰：Scum只是一个简单的沟通框架，所以大家的接受程度会比较高，只是规定了你该在什么时候沟通，该在什么时候反思，早上起来应该怎么跟大家交流，抽张扑克牌估算一下这个东西，Scum只是一个实施框架。极限编程是大量工程实践的集合，比如极限编程里讲TDD，它是具体让你操作的，比如咱俩就在一块，它是一种具体的工程实践，就是这么做。但具体工程实践大家反感就比较大了，因为跟他以前的做法不一样，而且问题的关键在于实施这种东西最重要的是有一个人带一个人去实施，两个人都不会你就在那儿极限编程，这个太扯了，根本一点用都没有，这种东西就是师傅带徒弟。

比如结对编程，两个人都没结对过，你如何在那儿结对？比如本来是一个大锯，你在这边我在那边，大家都没使过这个锯，咱俩就商量这个锯到底怎么拿，所以时间长了以后没有效果，大家马上就对这个不看好，然后就失败了，他们不能带来效果。一般在成熟的公司里面，他们是极限编程。就像结对这种最简单了，总共有两种，一种是师傅带徒弟，就是一个mentor，一个是徒弟在这里学，mentor一开始会给你演示所有的编程，你就在后面看，mentor可能会给你提问，等过了一阵你来，mentor在后面观察你、指导你，然后让你干这些。这是带新手非常有用的招，很快就能把新手培养出来。

另外一种就是两个人的水平差不多，就是互相干活，一个人负责设计，另一个人负责实现，这是真正的工程模式。那个是教学模式，你就要求两个人合作很长时间，比如我说的设计他不懂，这就玩完了，没法在一起。关于你说的极限编程，两个人没在一起合作过，或者两个人都没搭档过，你还在电脑上干自己的，后面的人就特别没意思，没有互动没有交流，好多工程实践都是这样的，虽然跟你平时的编程方法不一样导致的。Scrum跟大家多数的东西都不冲突，只是原来要开会，现在也要开会，现在规定好了什么时间开什么会，只是轻度地改变了你的东西，所以实施起来非常容易，大家的认可度也非常高，而且某种程度上见效又比较快。主要是因为它能发现一些具体的非技术上的问题，比如沟通的障碍、

理解需求有问题，或者节拍上有问题，有的部门太快了有的部门太慢了，他们能发现这些问题。

记者：当一个敏捷团队工作时，有时透明化的流程会暴露出机构中的问题，而这些问题又被称为敏捷开发流程的过失。在整个行业中，您们开始遇到这种情况了吗？敏捷开发会使行业的缺点逐步暴露，从而在各方面招致一些与敏捷开发对抗的反对意见吗？

程显峰：是这样的，按照我的理解，我觉得暴露问题是个好事，暴露问题就说明这不是一套有效的方法。为什么丰田要做单件流的系统呢？因为任何一个环节出了问题，马上就要停掉工作，他是可以以最快的速度发现问题、解决问题。为什么是一键而不是两键呢？因为一键可以发现的更及时，要求生产线上的所有环节都要严格匹配，如果有稍微不匹配的地方马上就会跳出来。如果它是一种发现问题的话，我觉得这种很好，不发现才是问题，发现问题不好吗？我觉得非常好。当然，你肯定会遭到反对意见，这个我觉得是实施的人应该想清楚的事情，实施什么会遭到反对意见，或者不实施什么会遭到反对意见。

记者：在实施当中有没有过这样的情况？

程显峰：肯定会有，这个可以来自于各个方面，但是你要实施的话就要想清楚，比如老板什么时候会反对？不出效益的时候会反对。所以刚开始实施的时候不会去做那些影响步骤的问题，你要先见效，有很多方法可以先见效，但是很多人都不知道，你要实施那些先见效的，这对树立信心、树立威望都是非常有帮助的。你不能一开始就挽起袖子说要怎么干，没有获得之前这些事情是做不了的。我们团队刚开始还好，有些团队你要实施敏捷，或者实施某些具体工程实践的话会有挑刺，这就跟你实施人员的素质有关了，比如你能说服他就说服，要是说服不了他的话你也没法实施。

有的人去实施版本控制，他就说要用分步式的传统模式，有些人就说为什么要用分步的？那么你做一个版本宏观动作，我用这个做一个，我们比一下。因为技术这个东西很好衡量，大家一看就都明白，它不比了，就实施吧。你要有手段和信心去实施这个东西，而且你要知道对手是什么，如果你控制不住这个局面的话，以后实施的时候会有各种各样的东西。比如实施测试，测试的好处你有没有给团队看到？你自己都不会测试，也带来不了什么好处，你就说服不了团队去做这种事情。

比如这里是有框架层面的东西，我特别愿意实施工程实践，因为我对工程实践的把控力特别强，你要是反对我，我有很多理由去说服你，而且非常容易见效，你要很快地赢得工程技术人员的信任。反而在实施这个项目时大家会很嘀咕，觉得这个家伙到底写没写过代码？是不是只会这些？因为它不涉及到任何技术细节。工程实践就不一样了，你说代码这么写不合适，那你就得说出不合适的道理，你要那么设计的话就是一种技术的对抗。其实遭到反对意见不光是敏捷遇到，干什么事情没有反对意见？没有反对意见的事大家早就做了，也不需要其他人。

记者：其实我想了解你们开发的反对意见。

程显峰：在我看来大部分的意见其实是一种固有思维的反应，就是他们固有模式的一种反应，以及固有的工作习惯不愿意改变的反应。问题是他没有见到你要实施这种东西好处的前提下会跳出来反对，你要做的事情是要证明这种东西会给

他带来利益。比如刚才说的那种，你用你的版本控制，我用我的版本控制，你自己本身有非常丰富的经验，你看到别人那么做就能很清楚地知道这样做肯定是简单或者容易的时候就不会比了。要是知道自己一定会赢的话肯定会比，肯定是他实施的新东西对他来讲是有好处的，但是他又不愿意付出改变的成本，又不愿意让大家明确地见到这种好处，因为团队的人要是都知道这种好处你就没法演下去了。所以很多事情要把它透明化，就是不存在那种边边角角的东西。

当然，阻力是各种各样的，最好的方式就是把这些东西都透明化，这些程序员相对来说还是非常讲道理的，因为跟机器打交道多了，有一说一，思维方式还是比较客观公正的，不会说这些东西明明是效率高的却跟你说不行，一般不会有这种情形。

记者：目前国内出现了很多敏捷教练的角色，敏捷的教练需要具备什么样的素质?

程显峰：我个人觉得不管你具备什么样的素质，你要能和开发团队打成一片，要能赢得开发团队的尊重才能做得下去，否则你天天讲，人家不信你，其实你什么事都做不了，你可以不懂技术，但是开发团队都很信你，这也OK，但是开发团队一般都有一种倾向，就是技术沙文主义，不懂技术的人是不能打交道的，你要是想真正在这个团队里混下去的话就得跟他们打交道。不同的团队当然不一样了，比如华为有敏捷实施的红头文件。

记者：这种规范开发人员的工作习惯，优化他们的习惯，教他们一些工程方法，这跟CTO的职能有什么区别？技术部门的Leader职能跟它有什么区别?

程显峰：比如丹峰就负责技术架构设计运营，我就负责培训，发现团队的问题并整合这些看上去非常琐碎的事情，还有一些其它培训方面的事情，他们是思变，是想着怎么去打这个仗，我是像个教官一样想着怎么提高他们的战斗力。他们只想怎么用这些人，然后打哪儿，这是他们要管的事情，我是教他们怎么使用武器，让他们训练有素，学会配合这些常规性的事情。

记者：对于未来几年敏捷开发的发展，您希望看到哪些新方向?有何建议?

程显峰：发展方面的问题我还真不知道，还是觉得有点落后，这个还需要积累吧。很长时间才会有一些改进，国内技术上落后的已经比较少了，反而是管理上落后的非常非常多。比如Facebook的崛起，大家都看到它用的是什么技术，很少有人看到Facebook在壮大的时候立马就从e-Bay挖了一个人给他们做工程方面所有的事情，他们从很早期的时候就从e-Bay挖了一个高管去做所有工程化的东西，很早就做了，非常有远见，这些事情很少有人知道。Facebook每年开了什么会，开发了什么框架他都知道，到了一定程度你就会发现它不是几个毛头小子一个Idea就可以发展起来的。e-Bay大概已经有二十多年，其实是经验非常丰富的人去做这个事情，并不是一个Idea就能行，现在好像不太注重这个方面。

其实在美国都是由非常资深的人去做工程化，而且这个市场还是比较稳定，真正去做这些东西的人还有一个相对比较稳定的流动环境，不会是从哪里突然冒出一个人就能做这些事情，或者是凭一个Idea就可以做。

记者：微软就有自己的工程研究院和微软研究院，也是不一样的。

程显峰：其实从开发的各种东西来说，类似微软、苹果、谷歌这些大公司，他们在工程管理上都有很多非常有经验的人，但是这些人在国内往往不讲这些。比如原来谷歌测试的负责人段念就是讲测试方面的稍微多一些，那个属于工程化的一部分，你就可以看到一些，多半也是做类似的工作，实际上他是一个非常有经验的做工程化的人。

记者：主要还是国内管理上的问题。

程显峰：对，软件研发的管理落后的不是一点半点，国内那种落后的方式跟美国六七十年代有一拼。其实现实就是这样，尤其是互联网公司还在觉得自己是什么高科技，其实工程管理的水平差得一塌糊涂，但是不是垄断高额利润，如果是从一个软件公司的角度去衡量的话，那个研究体系简直是一塌糊涂。

记者：其实从工程化的角度也会影响本身推出产品的效率或质量。

程显峰：其实工程化是一种内功，为什么IBM谁打他都不太怕？内功非常深厚体现在几个方面。

记者：IBM也比较开源。

程显峰：它的内功深厚也是有各种方面，首先就是知识产权方面保护得特别好，专利大户向来都是第一，第二就是在基础研究各个方面，工程化的思想，好多软件工程的思想，软件工程的方法，他们也很注重敏捷的一些东西。这就衍生了一个问题，为什么抄IBM的人都很惨、都死掉了呢？因为内功不够。IBM的管理方法非常重型，轻型的公司想抄都死掉了，为什么？因为你根本无法调动那么多的资源驾驭那么重型的方法，你的内功不行，就是这样。IBM原来那个case版本控制系统使用起来多负责啊，最后用的大部分都是电信系统，就是跟它有相同内功级别的人。他的客户都是金融、能源方面的，要是小公司用IBM做咨询，即便是你付得起钱也实施不下来。

记者：实施下来也达不到那种业务规模。

程显峰：我是觉得软件业的发展还是应该从基础做起，实实在在地把软件做好，不要过分地强调那些概念，好多人都是想解决别人的问题，每年实施的过程当中尤其要注意，作为一个团队的leader，你最清楚的是团队自身的问题，你也最容易解决团队自身的问题，要是每天都解决自身的问题的话一定是个特别好的教练，根本不需要去搭理别人，但可以借鉴别人的方法，看别人是怎么借鉴类似的问题。一定要着眼解决自身的问题，因为他们解决的都是别人的问题，比如看到别人的沟通方法就觉得自己的团队沟通有问题，实际上你就是三个人，你有什么沟通问题？根本就不存在。

记者：听你这样说，敏捷并不是一个特别有别于传统开发流程、软件开发工程的方法，只是在里面加入了一些管理的理念，能够把工程化和管理结合起来，可能就是一种敏捷。

程显峰：软件行业的管理方法主要来自于两个途径，六七十年代的时候大体上是借鉴建筑行业的系统化管理方法，主要是需求管理、建筑管理方面的内容，另一次改进就是在九十年代后期，由于丰田在制造业上的改进，就是精益方法，主要的改造就是发现问题及时生产及时交付，所有迭代周期这方面的东西。软件行业主要的来源就是制造业，实际上他们也不能太超越这两种。我就觉得现在老是想标榜自己跟别人不同，反而是

把自己给害了，那些方法实际上会在那些产业里实施得更好。

你看《精益软件开发艺术》那本书，写得特别好，那是波音的一个家伙写的，因为波音在制造业生产的时候用了这种方法，他在波音的软件部门，借鉴了波音的生产，然后再去用精益的思想改造，实际上都是借鉴其它传统行业的。传统软件的方法主要是建筑业的，建筑业和制造业都是比较成熟的，他们也会互相借鉴，借鉴最后的结果其实是发现方法和方法之间的差别没有想像的那么大，可能只是流派之分而已。比如大家都注重的版本控制、质量管理，整个周期管理这些东西，哪个没有呢？而且具体的操作方法大家都可以借鉴，你敏捷比较推荐的版本控制，实施到传统软件当中可不可以？也可以的。实际上并没有那么不同。我觉得这是一种商业策略，给划分了非黑即白，不是那样子的。其实大家真正在玩的时候就会觉得他们是可以有很多互通的，大家也可以互相借鉴，而且两种方法都在成长，这就跟武侠小说里的江湖差不多，大家都是为了各自的势力划分出来，他们就是邪教，我们就是明门正派。

其实那两个人讨论的就是一个风花雪月的问题，完全没有必要分什么派别，大家讨论的就是一个质量控制的问题，敏捷也可以用，传统也可以用，根本没有必要说这个方法就是属于谁的，但是这种实践的东西多起来之后就会发现越来越多的共同之处。

记者：最后就会发现自己在实践的过程中总结一套结合了传统和敏捷的方法。

程显峰：肯定的，我觉得比较负责、比较讲究实际的人都会结合自己这块东西去实施的，敏捷方面有个非常重要的东西，要是按照书本上实施的话肯定不是敏捷，最重要的就是裁减，你要按照自己的东西去裁减，包括传统的软件研发人员那一套也有裁减，只是很多人都不知道。传统的软件研发虽然很重，但是会按照自己的需求去裁减，要是完全按照那种东西就会死得很惨，所以你要实施一套流程的时候第一件事情就是裁减，要把它裁成合适自己那块，包括电信企业里面实施的东西都是经过裁减的。

原来我的一个同事他们在项目实施的时候首先就是要看标准的研发流程，弄下来一份之后再裁减，就说这个我们不要了，那个我们不要了，要是按照标准化他们也受不了，肯定不行，包括传统的实施。比如你就在一个猪圈，非得按照摩天大楼的质空标准去做，肯定不行，虽然都是建筑，但是不符合实际，所以最重要的就是符合实际产生效果，拿给老板也好说好看。

记者：那种大型的软件，比较像传统的，他们之间也有软件开发的部分吗？

程显峰：肯定会的，工程实践现在已经借鉴了很多，包括版本控制一般都是集中式的，现在大家对分布式的都会比较倾向，甚至是说发布流程。这些东西无所谓谁都可以用，比如传统上的需求管理软件、Bug跟踪系统，敏捷也在用。敏捷也有这些东西，你也需要跟踪Bug，你也需要有针对性地上线打包。

记者：所以只要技术能力强，跟敏捷没有多大关系，都是在创业团队里用敏捷的比较多。

程显峰：是这样的，大家为什么要用敏捷？很多都是创业小团队偷懒的行为，因为他实施不了那种重型的，也没有实施的经验。其实在国内实施规范化的还是相对较少，他们觉得这个东西能

够玩转才这么做的。而不是真正考察过，做了一个比较公正的技术选型，比如传统上我们会得93分，用了敏捷会得94分，根本没有数据，就是自己想，那种我们能不能玩？玩不了，这种我们能不能玩？差不多，实际上差得挺多的。因为敏捷强调的是不同，不同的东西又很少，所以大家就会觉得很容易。其实国内真正好的测试团队简直凤毛麟角，能玩得起测试的凤毛麟角，能把发布上线流程做好的也很少，这两个都做好了，我觉得你是敏捷还是传统都已经不重要了，这些对你才是实实在在的东西，对于质量、对于产品都是非常有益处的东西，至于长什么样子已经不重要了，关键是对你是否有好处。

大家现在都是有点炒概念，不讲实惠，我是比较讲实惠的，这些东西好不好我很快就能知道，所以我要的是实惠。我向老板汇报的时候也是这样，老板不看这些概念，他也不懂，创业的人好多是Idea出身，炒个概念会很好看，也会很好卖，但是大家现在都知道了，也不见得好卖。

记者：你们现在关注团队的开发，是从哪个方面去看这个问题？

程显峰：我个人比较关注的就是发现问题的方法，比如传统的东西，包括制造业的那些管理方法我也都会看一些，比如《约束法》、《关键目标》的那套东西，主要是强调管理的，各种各样的管理思想都能启发你，实际上并不需要看特别针对软件行业的东西，要看软件行业里的东西我一般比较愿意看失败的案例，就是类似《完美软件》那种常见的错误，比如程序员不善于沟通，然后说不是自己错误。人家在七十年代的时候已经把这些总结得特别好了，你会发现过了四十年还是那些问题，你就看那些老的书就会发现这些问题已经存在相当长的时间了，完全没有什么新鲜感。里面刻画的东西都是一样的，具体的问题还是很一样的，可能百分之九十的问题都是书上写过的，不会有太大的差异，只是人不一样、环境也不一样，问题还是差不多的。比如注重质量、需求发散，这些问题都是非常常见的。

记者：看来还是比较大块的问题，而且也不是软件行业专有的，各个行业其实都有这样的问题。

程显峰：但是你看软件行业的大师以前写过这些话就会觉得挺有意思的，当然，你还可以借鉴一些制造业的，我觉得都会帮助很大。

记者：还有管理方面的，只要是这个行业的都适用。

程显峰：其实这个点应该也差不多，我看腾讯的荣昊也做软件管理，他也是什么管理方面的书都看了。如果光看这个方面的话，因为都是需要互相借鉴的，要是想做得比较好的话还是需要看很多东西。确实有些东西大家都太强调了，软件行业的人都特别喜欢强调个性，其实管理的方法通用性还是比较高的。你看德鲁克写的那些书，每一条都是挺实在的，尤其是项目场景非常好。

周金根：个人敏捷的创立与详解Scrum会议

■ 在敏捷实践中，Scrum可以说是用于运行项目的框架，它基于敏捷的原则和价值。对于管理严格的项目团队，Scrum会议也会每天都在同一时间进行。通过每日Scrum 会议，团队成员之间可以彼此相互熟悉工作内容，充分了解项目进度，相互帮助解决问题。

在敏捷实践中，Scrum可以说是用于运行项目的框架，它基于敏捷的原则和价值。对于管理严格的项目团队，Scrum会议也会每天都在同一时间进行。通过每日Scrum 会议，团队成员之间可以彼此相互熟悉工作内容，充分了解项目进度，相互帮助解决问题。那么，在Scrum中的Sprint计划会议应当如何进行？记者采访了敏捷个人创立者，周金根老师跟网友们谈谈Scrum的实施与Sprint计划会议。

个人简介：周金根，个人成长教练，软件产品架构师，培训师，敏捷个人创立者和推广者。

以下内容为采访实录：

记者：请问你是如何接触到敏捷开发的？敏捷个人的创立是出于什么样的想法？

周金根：最早我们是采用RUP开发的（可以看看从IT方法论来谈RUP），我记得在一个产品设计阶段，我们使用Rose画了很多图，但是最终开发的时候其实没有人去看；更为重要的是，产品交付之后需求变更很多。这之后，我就开始关注软件全生命周期的方法、技术和工具，于是了解了敏捷。最早是XP，然后是Scrum。08年我去了一个新的项目组，这个组没有开发流程方法，也不知道如何做需求，于是我在这个组实践了一些新的方法，其中包括Scrum。在Scrum的实施过程中，我发现并不像最初想象的简单。看似简单的流程和角色，真正要发挥功效并不容易，除了技术能力之外，我还需要让自己学习更多管理方法，更需要自己深入领悟敏捷管理背后的东西。对Scrum的深入思考，我越发觉得团队中每个人的成长是敏捷发生效果的动力，这让我对个人管理这个话题有了更加浓厚的兴趣，于是把自己在个人管理和个人成长方面的思考写下来，并在团队中学习和实践。团队中的成员有自发打印出来给家属学习的，也有主动参与讨论的，以及后期社区对敏捷个人的认可，这都鼓励和激发我慢慢系统化的思考个人成长，并提出了敏捷个人。 经过两年多的思考，目前敏捷个人已经是一个较为体系的个人成长框架，它可以帮助个人成长，并促进敏捷团队的形成。

记者：请问敏捷开发是否真的能解决传统开发的一些问题？如何去认识敏捷的根本？

周金根：从瀑布到敏捷，我们已经发现产品更适合用户、质量越高、上市时间也越短，敏捷顺应不断变化的时代，是否能解决传统开发已经不是一个问题。对于敏捷，我认为其根本在于学习和适应，这也是拥抱变化需要具备的两项最重要的能力。

记者：团队的敏捷实践与管理是分不开的，您是怎么看目前国内的管理？

周金根：在没有敏捷之前我们就在管理，但在敏捷盛行的时候，有些管理者就分不清敏捷和管理了。其实我不太在意某种最佳实践出自哪种敏捷方法，我认为只要有利于当前团队的实践都是管理的一种工具，也就是说，作为管理者，我们仍需保持更大的视角，敏捷仅是管理的一种工具，我们还要学习目标设定、流程优化、团队建设、个人成长等更多管理方法。

记者：有些人认为敏捷开发并不适用于水平一般的程序员或团队，您是怎么认为的？

周金根：任何方法都不是银弹，也说明了没有一种方法是完美的。既然没有完美的方法，那自然也不必是完美的人去执行了。水平高的程序员

在技术实践领域的敏捷固然可以做得更好，但一个产品的失败大多数都不是因为你是否采用了测试驱动、结对编程等最佳实践，而是开发管理上的问题。Scrum作为一种敏捷方法，背后具有很多管理思想，只要管理者和团队对Scrum有进一步的思考和认识，也可以很大程度上去提高技术水平一般的程序员团队。

记者：敏捷方法在国内实施起会导致项目管理原有模式的改变，而很多公司都没有达到敏捷的目的。以致公司往往不愿意引进这样的开发模式。您怎么看待这个问题呢?

周金根：从公司角度来说，你能把软件越快越好的做出来就可以了，至于采用的是敏捷还是瀑布并不重要。与其说是公司不愿意引进这种模式，不如说是软件开发负责人不愿意或者没有能力引进新的方法。敏捷开发相对来说已经比较成熟，我认为现在不是讨论是否愿不愿意引进这种开发模式的时候，反而是思考如何引进的问题，这不仅仅是技术实践，还有管理，甚至是个人成长方面的改变。

记者：团队的人数对于敏捷开发有何影响？如何进行拆分?

周金根：人数的规模会带来团队的复杂性，随着人数增加，管理、沟通等都会越来越困难。而保持7±2人左右的小团队，可以更利于团队的形成。在这样的团队中，人与人之间都更为熟悉，协作起来就更容易。那这样的小团队由哪些人组成呢？这也需要根据产品的规模来定。对于一个小型产品，这个团队将由市场、需求、开发、测试等人员组成全功能性团队；如果产品属于中大型，那有可能会形成单一功能性团队，再由多个这样的团队组成一个大的敏捷团队，由这个大的团队来实现对客户的交付。

记者：敏捷实践过程中Scrum实施整个过程怎样规划。

周金根：实施Scrum，我们可以采用类似学习一样的过程，首先完全按照Scrum流程执行；然后再根据执行后的效果进行自我裁剪和补充；最后淡化Scrum的概念，与更大范围的软件产品周期过程融合起来。

记者：敏捷开发的方法内有很多不同的程度，而几乎每个敏捷开发团队都有scrum会议，在您们的团队中是如何进行的?

周金根：沟通在任何团队都是必不可少的，而会议是其中一种。我认为Scrum中的Sprint计划会议是最重要的事件，这确定了每次迭代的目标。回顾会议是第二重要的事件，因为这是团队做改进的最佳时机，如果没有回顾，就会发现团队在重犯相同的错误。如何进行可以看看我之前写的几篇blog：（以下内容摘自周金根博客）

1、Scrum之 Sprint计划会议

在sprint第一天召开sprint计划会议，这个会议分为两部分，计划会议1由PO、SM和Team参加，主要是从产品backlog中挑选出需要放到当前sprint下的既定产品backlog，然后由SM、Team参加计划会议2，把既定产品backlog的故事拆分成任务进行估算，PO也可以一起参加这个部分来了解具体的开发细节。以下我将把会议主要内容罗列。

会议内容

sprint计划会议1

产品负责人和团队一起，在先前评估的成果基础上，定出 Sprint 目标和既定产品Backlog。

目标

定出 Sprint 目标和既定产品 Backlog

会议准备

» 邀请与会者：产品负责人、Scrum Master、团队所有成员

» 已按优先级排列产品 Backlog 中各项问题

» 已评估 Backlog 中的各项问题

» 把产品 Backlog 公开给会议中的每个人，保证其可被获取

» 预期团队中有哪些人已明确会缺席(如度假)

» 保证房间环境适合小组讨论

» 每个人都可以获取上次 Sprint 评审会议和 Sprint 回顾会议的结果

1. Sprint 时间表已经安排
2. Sprint 计划会议 1 的时间安排
3. Sprint 计划会议 2 的时间安排
4. Sprint 的第一天已确定
5. Sprint 的最后一天已确定
6. Scrum 每日例会的时间安排
7. Sprint 评审会议的时间安排
8. Sprint 回顾会议的时间安排
9. （可选）为既定 Backlog 准备图钉板:一个至少 2x2 米的图钉板、卡片和贴纸、荧光笔
10. （可选）用作计划纸牌的卡片

会议进程（4 小时）

» 把 Sprint 时间表公开给所有人

» 把 Sprint 评审会议的结果公开给所有人

» 把 Sprint 回顾会议的结果公开给所有人

» 产品负责人向团队产品阐述产品远景

» 产品负责人和团队一起确定 Sprint 目标

» 如果 Backlog 里有问题遗漏：产品负责人有权限往 Backlog 里添加问题

» 如果产品 Backlog 完全未被评估：选择 Backlog 中您认为是最小用例的问题，并指派其工作量为 2 个Story Point。以这个最小用例的工作量标准，分配 Backlog 中其他问题的 Story Point

» 如果 Backlog 中的一些问题尚未被评估：根据其他问题工作量，评估这些问题的 Story Point 量

» 如果产品 Backlog 中的各项还没能合理地按优先级排序:产品负责人对产品 Backlog 中的各项按优先级排序

» 产品负责人和小组成员相互认可这 Sprint 目标和既定产品 Backlog

会议结果

为Sprint计划会议2的进行准备好既定产品 Backlog

sprint计划会议2

在 Sprint 计划会议 2 中，团队将既定产品 Backlog 中的每一项细化成多个任务。每个任务完成的时间限定在一天内。

目标

确定所有任务，生成 Sprint Backlog，确认 Sprint 目标

会议准备

» 邀请与会者：Scrum Master、团队所有成员、产品负责人（可以有权得知所有问题）

» 任务规划时可以参考既定产品 Backlog

» （可选）为既定 Backlog 准备图钉板:一个至少 2x2 米的图钉板、卡片和贴纸、荧光笔

会议进程（4 小时）

» 团队成员从 Backlog 的各项问题中分出相应的任务

» 确保考虑到工作中所有的细节:编码、测试、代码评审、会议、学习新技术、编写文档

» 如果任务需时超过一天：尝试把该任务分割成几个小任务

» 如果团队认为 Sprint Backlog 中项过多：和产品负责人一起删减 Backlog 中的问题

» 如果团队认为 Sprint Backlog 中的项过少：和产品负责人一起从产品 Backlog 中选出最重要问题，加入Sprint Backlog 中

» 团队确认 Sprint 目标

会议结果

» Sprint 目标和 Sprint Backlog 对于公司内的所有人都是公开的

» 所有团队成员都可以获取 Sprint Backlog 中的任务

2、Scrum之 站立例会

在sprint期间，每天都会通过站立例会来进行沟通，以下我将把会议主要内容罗列一下。（以下会议内容来自于Scrum Checklists)

会议内容

目标

团队成员间工作进度的沟通和协调

会议准备

» 邀请与会者：团队所有成员、Scrum Master、产品负责人（可选）、相关人员（可选）

» 在Sprint Backlog 上的所有任务都是可以增删修改，可重排序的

» 任务的状态可设为todo、doing、done（可以再加一个test表示需要验证）

会议进程（15 分钟内）

» 上次会议时的任务哪些已经完成：把任务从“正在处理”状态转为“已完成”状态

» 下一次会议之前，你计划完成什么任务？
如果任务状态为“待处理”：转为“正在处理”状态

1. 如果任务不在 Sprint Backlog 上：添加这个任务

2. 如果任务不能在一天内完成：把这任务细分成多个任务

3. 如果任务可以在一天内完成：把任务状态设为“正在处理”

4. 如果任务状态已经是“正在处理”：询问是否存在阻碍任务完成得问题

» 有什么问题阻碍了你的开发：如果有阻碍你开发进度的问题，把该障碍加入到障碍 Back-log 中

» 如果展开了一个问题的讨论：提醒团队的成员们注意把精力集中在回答关键问题上

» 如果相关人员想发表些言论：礼貌地提醒他，该会议只允许让小组成员讨论

会议结果

» 得到最新的障碍 Backlog

» 得到最新的 Sprint Backlog

» 最新的工作进度图

其他

可以指定一个主持人（或轮流）。他来召集并控制会议时间，会议中注意引导话题，在会议结束时可以做个简短的总结，说出重点就行，做好每日规划。

3、Scrum之 评审会议

在sprint周期最后，需要进行一次评审会议，让团队向产品负责人和利益相关者展示已完成的功能。sprint审核的大部分实践用于团队成员展示功能、回答利益相关者对展示的疑问并记录所期望的更改。评审会议可以吸引相关利益者的关注，让其他人了解团队在做些什么，并得到重要反馈。做演示也会迫使开发团队真正完成一些工作。

» 小组准备好工作站和设备等等，用以展示产品的新功能

» 团队准备sprint审核实践不应超过1小时

会议进程（4小时）

» 确保所有人员都清晰目标，如果有人对产品不知道，则花几分钟来进行描述。

» 团队按 Backlog 中的问题，逐个地介绍这次 Sprint 的结果，和演示新功能。

» 如果产品负责人想要改变功能：添加一个新问题到产品 Backlog 中

» 如果对功能有一个新的想法：添加一个新问题到产品 Backlog 中

» 如果小组报告项目遇到阻碍现在还没能解决：把该障碍加入到障碍 Backlog

» 会议结束时，ScrumMaster向产品负责人和全体利益相关者宣布下一次审核的地点和时间。

会议结果

» 对这次 Sprint 的结果和整个产品的开发状态的共识

其他

» 让演示关注业务层次，不要关注技术细节。注意力放在“我们做了什么”，而不是“我们怎么做的”

» 有的sprint可能会包含很多bug修复等功能，在评审会议中不要演示太多一大堆细碎的bug修复，除非这个很重要。

4、Scrum之 回顾会议

Scrum中Sprint计划会议是最重要的事件，第二重要的事件就是回顾会议，因为这是团队做改进的最佳时机。如果没有回顾，就会发现团队在重犯相同的错误。在sprint的评审会议后，团队需要进行一次回顾会议，以下我将把会议主要内容罗列一下。（以下会议内容来自于Scrum Checklists和scrum-and-xp)

会议内容

目标

通过总结以往的实践经验来提高团队生产力。

会议准备

» 邀请与会者： Scrum Master、团队所有成员、产品负责人（可选）

» 附属工具：为所有参与者准备的荧光笔、贴纸、白板磁吸、白板和挂纸板

» 准备一个回顾白板，分三列。第一列和第二列是回顾过去，第三列是展望将来。

1、Good：如果重做同一个sprint，哪些做法可以保持

2、Could have done better：如果重做同一个sprint，哪些做法需要改变

3、Improvements：有关将来如何改进具体想法

会议进程（1－3小时）

» 介绍会议目标和议程

» 准备（setting the stage）：制定和回顾团队价值观和约定（Team values and working agreements）：(10−30分钟)

1、不管我们现在发现了什么问题，我们必须懂得并坚信每个人通过他们当时所知的，他所拥有的技能和可得到的资源，在限定的环境下，都尽其所能做出了最好的成绩

2、每个人都参与

3、坦诚交流

4、多说I，少用You

5、不深究具体业务细节内部问题

» 收集数据（Gather Data）：收集硬数据：事件（events）、度量（metrics）、完成的故事等

1、事件：对团队每个人都重要的任何事件，包含会议、决策点、里程碑、采用新技术等，例如上期回顾会议中确定的改进项是否执行了

2、度量：包含燃烧图、速度、bug数、完成故事点数、代码重构数等

» 产生见解（Generate Insights）：多问“为什么”，从收集的数据中找出优点和问题

1、向与会者解说如何使用该贴纸进行工作：使用贴纸时，注意一张贴纸只记录一件事。

2、派发贴纸，通过头脑风暴分别得出回顾白板三列的所有想法。

» 确定改进项（Decide What to Do）

1、每人三票，投票决定下一sprint着重进行哪些改进（2－5项左右）。

» 结束回顾（Close the Retrospective）

1、给会议做个总结，表明下一个回顾会议需要跟踪哪些做法

会议结果

回顾白板，以及下一sprint需要改进的做法，在下一个回顾中，会跟踪这些改进的执行情况把障碍增加到障碍 Backlog 中去

记者： 对于未来几年敏捷开发的发展，您希望看到哪些新方向?有何建议?

周金根：敏捷只是一个代名词，我希望它不仅仅只包含开发，还能能够在基于敏捷思想下，把方法框架扩充到市场、业务、营销环节等软件产品开发全生命周期。

编者按

我们都知道敏捷是一套流程和方法的持续改进，它能够通过快速迭代的方式交付产品。而这个抽象的流程形式在不同项目中会根据具体的项目特征进行裁剪，这涉及到了团队在Scrum中实施过程的规划与拆分的问题，下面记者采访了Thomson Reuters公司流程改进部门经理陈斌老师，给网友们分享了敏捷在项目中实施的一些主要问题。

陈斌：Scrum实施规划与团队拆分

我们都知道敏捷是一套流程和方法的持续改进，它能够通过快速迭代的方式交付产品。而这个抽象的流程形式在不同项目中会根据具体的项目特征进行裁剪，这涉及到了团队在Scrum中实施过程的规划与拆分的问题，下面记者采访了Thomson Reuters公司流程改进部门经理陈斌老师，给网友们分享了敏捷在项目中实施的一些主要问题。

个人简介：陈斌老师目前任职Thomson Reuters公司流程改进部门经理，负责敏捷教练（深入开发团队）与工具环境支持（包括工具开发及方案提供）。

10年产品/软件开发，以及8年质量保证/流程改进经验。经历过无流程，重流程，轻流程的各种风格。近年来的专业领域：参与CMMI评估以全面了解软件组织成熟度，引入敏捷原则及方法以提供应对业务挑战的具体变革方案，应用Six Sigma方法以保证流程改进产出业务价值。

以下是采访实录：

记者：敏捷实践过程中Scrum实施整个过程怎样规划。

陈斌老师：实施Scrum过程的第一件事情就是要明确为什么实施Scrum。目标应该是业务导向的，即解决什么痛点，预期的收益是什么，而不是敏捷转型。通过这样的分析，甚至可能意识到其他选择更为适合。例如产品已进入运维期，Kanban或其他方法更适合快速响应客户反馈。

如果确定了实施目标，需要把信息明确传递给每个工作人员，作为大家共同的目标，而不仅仅是某个“工作组”的目标。

其他实施过程包括：选择试点产品组，初期培训，框架设立，工具选择，组内教练，总结调整，结果度量，带动更多产品组。选择试点时要小心，如果试点组与其他组依赖关系很强，成功难度很高。

记者：迭代开发过程的一些困难与解决方法。

陈斌老师：最大的难点在于每个迭代都产出最有价值的产品增量，并且是有质量保证的。

前者要求能从重多需求中把握到客户核心价值，并分解到短期可完成的完整的用户故事。产品经理的选择是重点。

后者要求评审，集成与测试基本同步于开发完成。测试不仅是新功能测试，还包括回归功能测试/性能测试等。持续集成与自动化测试由此会成为高优先级任务，而不是在传统开发模式中的“nice to have”。

记者：敏捷实践中团队的拆分是如何进行的?

陈斌老师：首先认清事实，同样是合作，跨组合作远远难于组内合作，无论管理层如何强调。所以拆分的重点在于如何减少不必要的跨部门合作。“一站式”服务团队是目标：从需求到上线，绝大部分环节在组内完成，协调沟通发生在组内。

当产品规模很大时（例如团队超过100人），每个独立团队很难做到“什么都能做”，合理的解耦是关键。能够落实到任一用户功能能由一个团队独立完成（或很少的外部依赖）就可以了。

记者：如何更有效的做好开发流程估算?

陈斌老师：初始估算只能先“拍脑袋”，在迭代中参照实际调整估算，这是迭代开发的优势之一。重点：发布计划不能变，重大功能点不能变，具体需求细节随时调整。

记者：敏捷实践下的程序员工作指标将如何衡量?

陈斌老师：先衡量团队绩效。成熟的指标包括产品发布版本的缺陷数目，上线时间，与竞争对手的比较。

在团队内部360度互评贡献度。

记者： scrum会议，在您们的团队中是如何进行的?

陈斌老师：管理者退一点儿，团队成员进一点儿。

记者：团队中是否有敏捷测试人员？需要哪些技能?

陈斌老师：有。一方面强调团队整体对质量负责，对测试负责；另一方面在团队真正成熟前，专职的测试人员能保证“测试”任务不被功能任务挤掉，同时测试人员的独立视角很有帮助。

全功能团队不是说每个人都作同样的工作，而是没有职能“边界”与灰色地带。

记者：敏捷教练在Scrum中如何进行角色转换?

陈斌老师：能进而作为角色榜样，而目标在于退出团队角色时团队运转良好并在自我优化。

记者：对于未来几年敏捷开发的发展，您希望看到哪些新方向?有何建议?

陈斌老师：迭代开发，持续集成等已深入人心。难点在两头，也是努力的方向：组织结构（包括文化，部门划分，绩效考评）和深入的工程实践（例如测试开发同步）。

“敏捷个人”实践活动PPT讲义【周金根】

敏捷Scrum常被用于敏捷开发，敏捷管理，但是周金根先生（jingen_zhou）尝试将敏捷一词用于个人的学习和管理，并提出“敏捷个人”的概念。本专题收录了他在敏捷个人历次实践活动的PPT讲义资料，在此特别鸣谢！

敏捷个人实践第14次活动：学会感恩.pptx

敏捷个人实践第13次活动：获得他人的反馈

敏捷个人实践第15次活动：编写生活手册

第1次线下：敏捷个人的成长思考

敏捷个人2011.6月线下活动PPT：知道力、战胜拖拉

敏捷个人实践第12次活动：整理你的空间

敏捷个人实践第2次活动：生活方向盘 PPT以及活动录音

敏捷个人实践第16次活动：成为早起者

敏捷个人实践第17次活动：布置自己的工作间

敏捷个人第2次线下活动PPT：习惯、思维导图、活动照片

敏捷个人－认识自我，管理自我 v0.2

有道李勤飞：敏捷不是快速，更多的是灵活

■ 编者按

敏捷开发从国外引进发展已经有了很长的一段时间，而在国内的追棒依然很火爆，中小企业，创业公司很多项目成员开始学习敏捷，采用以及转向敏捷开发。可是，当实际操作上并不能解决……

敏捷开发从国外引进发展已经有了很长的一段时间，而在国内的追棒依然很火爆，中小企业，创业公司很多项目成员开始学习敏捷，采用以及转向敏捷开发。可是，当实际操作上并不能解决传统开发的一些问题解决，反而新增加了更多的问题。

有人说，实践敏捷的根本不在于敏捷本身，而在于理解敏捷背后拥抱变化的基因。确实，使用敏捷，那么你就应该知道为何敏捷。

当你从某个行业去领悟衍生出这种方式，毕竟那是成熟行业成功的经验映射。在实际的操作中，分配，转型，时间，技能等等都需要严格谨慎的执行。

就在前段时间，网易有道云笔记负责人蒋炜航在微博上分享了一篇名为《敏捷开发的实战经验》的文章，讲解了团队如何实践scrum的一些经验得到了网友很高的评价。网易有道云笔记从事敏捷开发积累了丰富的经验，因此，51CTO的记者基于敏捷开发为主题，采访了网易有道研发经理李勤飞。

李勤飞目前是网易有道研发经理，有道云笔记技术负责人。曾参与过有道词典的开发，具有多年的团队管理经验。是有道云笔记引入和实践敏捷开发方式的主要负责人。

以下是李勤飞给网友们分享的敏捷开发管理经验：

一、敏捷团队

1）在敏捷开发中团队的拆分

在敏捷开发中，组织团队方式是按照项目组织，而不是行政划分。拆分团队只有一个理由，就是能提高团队效率。根据经验，小团队的效率更高。当团队人数大于9个时，计划会和站会，成员的参与感会下降，效率会降低，沟通成本也会加大，这时候需要拆分团队。

2）在敏捷开发中团队的管理

敏捷开发只适用于小规模的团队的这种说法是错误的，敏捷开发跟团队的数量没有直接关系，只要大于3人的团队都可以实行敏捷开发。

但是，小团队更容易敏捷。团队人数的增加必然会提升沟通协作的成本，可以通过拆分团队的方式来提升效率。把一个大团队拆分成几个小团队，团队之间的协作也走敏捷开发的流程，就是我们常说的Scrum of Scrums。

3）在敏捷开发中团队的转型

想要做到敏捷开发，每个团队都要经历一个转型期，那么在转型期还需要每个团队根据自身的不同，找出合理有效的解决方法。

对于有道云笔记的团队是逐步推行敏捷开发的，没有遭遇明显的转型期，但推行过程中确实也遇到一些问题。比如产品经理开始并不认同根据固定时间的sprint来划分版本，最后用已有团队整体产出提升的经验说服了他。还有比如产品已经上线，sprint进行中间会有一些线上的问题插进来，我们通过线上值日，以及根据经验数据来预留时间等方式来解决这个问题。

二、敏捷方式

1）编码方式

很多人为了编写易变更的代码，在采用敏捷时，抛弃许多习惯用法，这是一个误解。敏捷开发推崇涌现式设计，通过不断的重构来实现更好

的架构。指的设计而不是编码，对于编码方式，不需要发生变化，除了需要遵守公司和团队的编码规范外，用自己熟悉的方式即可。

2）极限编程（XP）

极限编程有很多争议，我们的方式是选择性接受，比如把两位工程师组成一对，相互review代码，并且要求编码测试代码等。但是暂时不会采用两人一起编程的方式。

3）指标衡量

每个sprint的总结非常重要，会记录每个任务（task）的预估时间和完成时间。并且定义了完成度（任务的完成情况），希望sprint的完成度在80%以上。如果完成度低，就需要总结原因并改进，团队成员的绩效也会受影响。当然，除了项目进度以外，有道公司还有另一套评价体系，跟业务无关，而是由专业的技术委员会对程序员个人能力的评估。这两套评估结合在一起，就可以很好的衡量程序员的工作。总的来说就是，按进度完成项目的同事，个人能力也需要不断提升。

4）scrum会议

Sprint会议是实践scrum中最重要的事件，这更是团队做改进的最佳时机。有道云笔记团队的Sprint以一个月为期，四种会议：需求梳理会，Sprint计划会，Scrum例会，Sprint回顾会议。

需求梳理会在Sprint计划会的前1-3天开，参加的人员是产品经理和开发人员，由产品经理称述需求，开发人员讨论设计与实现。

Sprint计划会是Sprint开始第一天开，参加的人员有产品、开发和测试，由产品经理讲需求，开发人员把需求分解成任务。

Scrum例会每周两次，周二和周四，主要沟通任务的完成情况，和下一个两天做什么。

Sprint回顾会议在Sprint结束时做，主要总结这个Sprint的经验教训，并且达成一些团队共识，比如例会迟到如何惩罚等。

最后，

李勤飞希望未来敏捷开发在多地方点办公，敏捷测试等方面有新的发展。建议大家使用敏捷开发时都可以做些调整，然后把自己的实践经验分享出来，共同改进这套方法。

资料下载

» 中国敏捷软件开发成功实践案例集

» Scrum--敏捷开发过程框架介绍

» SAP BUSINESS ONE中文版培训教程-敏捷系统管理

» Ruby on Rails敏捷开发实践源代码

» 敏捷开发项目管理IBM的PPT

» 敏捷建模：极限编程和统一过程的有效实践

» 使用Rails进行敏捷web开发-第四版beta8

» 《应用Yii1.1和PHP5进行敏捷Web开发》（中文版）

» SAP+B+ONE++敏捷库存+pdf+教程

» Oracle Application Express敏捷开发

敏捷开发及其给业务发展带来的影响

□ 布加迪/译

敏捷开发方法对商业界来说并不是什么新概念；不管贵公司从事什么样的行业，如果贵公司想经济高效地发展业务，就需要考虑敏捷开发方法。敏捷开发方法有时对企业来说是最佳之道。如果公司里面的每个人都知道大家在干什么，那么完成任务就要容易得多、快速得多。在这样一种环境下，学习起来也很容易；公司上下几乎每个人都乐于接受变化和变革。

提高客户满意度

如果贵公司实施了敏捷开发方法，就很容易为你的客户拿出解决方案。而且客户满意度大大提高了，因为公司里面的每个人都在同一个层面上，他们知道某个客户到底想要什么。为客户提供解决方案是一回事，而确保客户满意是另一回事。有了敏捷开发方法，客户感到满意的可能性就要大得多。

缩短交付时间

采用敏捷开发方法，发展业务成了一件容易得多的任务，因为你不仅了解客户的视角，设身处地为客户考虑，还了解使用某一种业务开展方式是不是存在什么缺点和不足。从某个方面来看，这有助于引导你为客户提供最佳的解决方案。另外，还大幅缩短了响应时间。

提出合适的问题

说到敏捷开发方法，发展业务变得更容易了，因为拜访客户的那个人能够考虑到各个视角，提出合适的问题。提出了合适的问题后，你用不着烦扰客户，就能获取在平常的业务开发环境下原本可能会错失的信息。

与客户加强面对面交流

说到业务开发，你会有大量的时间需要与客户进行联系和交流。借助敏捷开发方法，你能够与客户更顺畅地交流，因为你能够以一种更有效的方式分析需求。此外，客户与明白需求的公司代表进行交谈时，对贵公司也抱有更大的信心。

提高满意度，缩短响应时间

敏捷开发方法最大的优点在于，客户没必要为拿到所需的产品等待很长的时间。此外，借助敏捷开发方法，交付的产品根本不会出现不一致的地方。如果客户在比较短的时间内拿得了自己真正需要的产品，当他们需要下一个产品时，更有可能会主动找上门来。

更容易适应困难情形

借助敏捷开发方法，发展业务也来得更容易，因为所有人碰到过不同的情形，因而拜访潜在客户时胸有成竹。他们会知道情形到底会往哪个方向发展，他们都能够把谈话内容引往正确的方向。

采用敏捷开发策略对公司企业来说是一种极其有效的方法。采用这种策略的公司取得的成功比其他任何公司要大得多。

英文原文：http://www.socialhunt.net/blog/agile-business-development/

封面设计：@51CTO小林

开发月刊（副刊）

51CTO开发频道